Using Bag-of-Concepts to Improve the Performance of Support Vector Machines in Text Categorization

Magnus Sahlgren
SICS
Box 1263
SE-164 29 Kista
Sweden
mange@sics.se

Rickard Cöster
SICS
Box 1263
SE-164 29 Kista
Sweden
rick@sics.se

Abstract

This paper investigates the use of concept-based representations for text categorization. We introduce a new approach to create concept-based text representations, and apply it to a standard text categorization collection. The representations are used as input to a Support Vector Machine classifier, and the results show that there are certain categories for which concept-based representations constitute a viable supplement to word-based ones. We also demonstrate how the performance of the Support Vector Machine can be improved by combining representations.

1 Introduction

Text categorization is the task of assigning a text[1] to one or more of a set of predefined categories. As with most other natural language processing applications, representational factors are decisive for the performance of the categorization. The incomparably most common representational scheme in text categorization is the Bag-of-Words (BoW) approach, in which a text is represented as a vector $\vec{t}$ of word weights, such that $\vec{t_i} = (w_1...w_n)$ where w_n are the weights (usually a $tf \times idf$-value)[2] of the words in the text. The BoW representation ignores all semantic or conceptual information; it simply looks at the surface word forms.

There have been attempts at deriving more sophisticated representations for text categorization, including the use of n-grams or phrases (Lewis, 1992; Dumais et al., 1998), or augmenting the standard BoW approach with synonym clusters or latent dimensions (Baker and McCallum, 1998; Cai and Hofmann, 2003). However, none of the more elaborate representations manage to significantly outperform the standard BoW approach (Sebastiani, 2002). In addition to this, they are typically more expensive to compute.

What interests us in this paper is the difference between using standard BoW and more elaborate, concept-based representations. Since text categorization is normally cast as a problem concerning the *content* of the text (Dumais et al., 1998), one might assume that looking beyond the mere surface word forms should be beneficial for the text representations. We believe that, even though BoW representations are superior in most text categorization tasks, concept-based schemes *do* provide important information, and that they can be used as a supplement to the BoW representations. Our goal is therefore to investigate whether there are specific categories in a standard text categorization collection for which using concept-based representations is more appropriate, and if combinations of word-based and concept-based representations can be used to improve the categorization performance.

In order to do this, we introduce a new method for producing concept-based representations for natural language data. The method is efficient, fast and scalable, and requires no external resources. We use the method to create concept-based representations for a standard text categorization problem, and we use the representations as input to a Support Vector Machine classifier. The categorization results are compared to those reached using standard BoW representations, and we also demonstrate how the performance of the Support Vector Machine can be improved by combining representations.

[1] In the remainder of this paper, we use the terms "text" and "document" synonymously.

[2] The $tf \times idf$ measure is a standard weighting scheme, where tf_i is simply the frequency of word i in the document, and idf is the *inverse document frequency*, given by $\frac{N}{n_i}$ where N is the total number of documents in the data, and n_i is the number of documents in which word i occurs. The most common version of the $tf \times idf$ formula is $w_i = tf_i \times log\frac{N}{n_i}$ (Baeza-Yates and Ribeiro-Neto, 1999).

2 Bag-of-Concepts

The standard BoW representations are usually refined before they are used as input to a classification algorithm. One refinement method is to use *feature selection*, which means that words are removed from the representations based on statistical measures, such as document frequency, information gain, χ^2, or mutual information (Yang and Pedersen, 1997). Another refinement method is to use *feature extraction*, which means that "artificial" features are created from the original ones, either by using clustering methods, such as distributional clustering (Baker and McCallum, 1998), or by using factor analytic methods such as singular value decomposition.

Note that feature extraction methods also handle problems with synonymy, by grouping together words that mean similar things, or by restructuring the data (i.e. the number of features) according to a small number of salient dimensions, so that similar words get similar representations. Since these methods do not represent texts merely as collections of the words they contain, but rather as collections of the *concepts* they contain — whether these be synonym sets or latent dimensions — a more fitting label for these representations would be *Bag-of-Concepts* (BoC).

3 Random Indexing

One serious problem with BoC approaches is that they tend to be computationally expensive. This is true at least for methods that use factor analytic techniques. Other BoC approaches that use resources such as WordNet have limited portability, and are normally not easily adaptable to other domains and to other languages.

To overcome these problems, we have developed an alternative approach for producing BoC representations. The approach is based on *Random Indexing* (Kanerva et al., 2000; Karlgren and Sahlgren, 2001), which is a vector space methodology for producing *context vectors*[3] for words based on cooccurrence data. The context vectors can be used to produce BoC representations by combining the context vectors of the words that occur in a text.

In the traditional vector space model, context vectors are generated by representing the data in a cooccurrence matrix F of order $w \times c$, such that the rows F_w represent the words, the columns F_c represent the contexts (typically words or documents[4]), and the cells are the (weighted and normalized) cooccurrence counts of a given word in a given context. The point of this representation is that each row of cooccurrence counts can be interpreted as a c-dimensional context vector $\vec{w}$ for a given word.

In the Random Indexing approach, the cooccurrence matrix is replaced by a context matrix G of order $w \times k$, where $k \ll c$. Each row G_i is the k-dimensional context vector for word i. The context vectors are accumulated by adding together k-dimensional *index vectors* that have been assigned to each context in the data — whether document, paragraph, clause, window, or neighboring words. The index vectors constitute a unique representation for each context, and are sparse, high-dimensional, and ternary, which means that their dimensionality k typically is on the order of thousands and that they consist of a small number of randomly distributed $+1$s and -1s. The k-dimensional index vectors are used to accumulate k-dimensional context vectors by the following procedure: every time a given word occurs in a context, that context's index vector is added (by vector addition) to the context vector for the given word.

Note that the same procedure will produce a standard cooccurrence matrix F of order $w \times c$ if we use unary index vectors of the same dimensionality c as the number of contexts.[5] Mathematically, the unary vectors are orthogonal, whereas the random index vectors are only *nearly* orthogonal. However, since there are more nearly orthogonal than truly orthogonal directions in a high-dimensional space, choosing random directions gets us sufficiently close to orthogonality to provide an approximation of the unary vectors (Hecht-Nielsen, 1994).

The Random Indexing approach is motivated by the Johnson-Lindenstrauss Lemma (Johnson and Lindenstrauss, 1984), which states that if we project points into a randomly selected subspace of sufficiently high dimensionality, the

[3] Context vectors represent the distributional profile of words, making it possible to express distributional similarity between words by standard vector similarity measures.

[4] Words are used as contexts in e.g. Hyperspace Analogue to Language (HAL) (Lund et al., 1995), whereas documents are used in e.g. Latent Semantic Indexing/Analysis (LSI/LSA) (Deerwester et al., 1990; Landauer and Dumais, 1997).

[5] These unary index vectors would have a single 1 marking the place of the context in a list of all contexts — the nth element of the index vector for the nth context would be 1.

distances between the points are approximately preserved. Thus, if we collect the random index vectors into a random matrix R of order $c \times k$, whose row R_i is the k-dimensional index vector for context i, we find that the following relation holds:

$$G_{w \times k} = F_{w \times c} R_{c \times k}$$

That is, the Random Indexing context matrix G contains the same information as we get by multiplying the standard cooccurrence matrix F with the random matrix R, where RR^T approximates the identity matrix.

3.1 Advantages of Random Indexing

One advantage of using Random Indexing is that it is an *incremental* method, which means that we do not have to sample *all* the data before we can start using the context vectors — Random Indexing can provide intermediary results even after just a few vector additions. Other vector space models need to analyze the entire data before the context vectors are operational.

Another advantage is that Random Indexing avoids the "huge matrix step", since the dimensionality k of the vectors is much smaller than, and not directly dependent on, the number of contexts c in the data. Other vector space models, including those that use dimension reduction techniques such as singular value decomposition, depend on building the $w \times c$ cooccurrence matrix F.

This "huge matrix step" is perhaps the most serious deficiency of other models, since their complexity becomes dependent on the number of contexts c in the data, which typically is a very large number. Even methods that are mathematically equivalent to Random Indexing, such as random projection (Papadimitriou et al., 1998) and random mapping (Kaski, 1999), are not incremental, and require the initial $w \times c$ cooccurrence matrix.

Since dimension reduction is built into Random Indexing, we achieve a significant gain in processing time and memory consumption, compared to other models. Furthermore, the approach is scalable, since adding new contexts to the data set does not increase the dimensionality of the context vectors.

3.2 Bag-of-Context vectors

The context vectors produced by Random Indexing can be used to generate BoC representations. This is done by, for every text, summing the (weighted) context vectors of the words that occur in the particular text. Note that summing vectors result in tf-weighting, since a word's vector is added to the text's vector as many times as the word occurs in the text. The same procedure generates standard BoW representations if we use unary index vectors of the same dimensionality as the number of words in the data instead of context vectors, and weight the summation of the unary index vectors with the idf-values of the words.[6]

4 Experiment Setup

In the following sections, we describe the setup for our text categorization experiments.

4.1 Data

We use the Reuters-21578 test collection, which consists of 21,578 news wire documents that have been manually assigned to different categories. In these experiments, we use the "ModApte" split, which divides the collection into 9,603 training documents and 3,299 test documents, assigned to 90 topic categories. After lemmatization, stopword filtering based on document frequency, and frequency thresholding that excluded words with frequency < 3, the training data contains 8,887 unique word types.

4.2 Representations

The standard BoW representations for this setup of Reuters-21578 are 8,887-dimensional and very sparse. To produce BoC representations, a k-dimensional random index vector is assigned to each training document. Context vectors for the words are then produced by adding the index vectors of a document to the context vector for a given word every time the word occur in that document.[7] The context

[6] We can also use Random Indexing to produce *reduced* BoW representations (i.e. BoW representations with reduced dimensionality), which we do by summing the weighted random index vectors of the words that occur in the text. We do not include any results from using reduced BoW representations in this paper, since they contain more noise than the standard BoW vectors. However, they are useful in very high-dimensional applications where efficiency is an important factor.

[7] We initially also used word-based contexts, where index vectors were assigned to each unique word, and context vectors were produced by adding the random index vectors of the surrounding words to the context vector of a given word every time the word ocurred in the training data. However, the word-based BoC representations consistently produced inferior results compared to the document-based ones, so we decided not to pursue the experiments with word-based BoC representations for this paper.

vectors are then used to generate BoC representations for the texts by summing the context vectors of the words in each text, resulting in k-dimensional dense BoC vectors.

4.3 Support Vector Machines

For learning the categories, we use the Support Vector Machine (SVM) (Vapnik, 1995) algorithm for binary classification. SVM finds the separating hyperplane that has maximum *margin* between the two classes. Separating the examples with a maximum margin hyperplane is motivated by results from statistical learning theory, which states that a learning algorithm, to achieve good generalisation, should minimize both the empirical error and also the "capacity" of the functions that the learning algorithm implements. By maximizing the margin, the capacity or complexity of the function class (separating hyperplanes) is minimized. Finding this hyperplane is expressed as a mathematical optimization problem.

Let $\{(\vec{x}_1, y_1), \ldots, (\vec{x}_l, y_l)\}$ where $\vec{x}_i \in R^n, y_i \in \pm 1$ be a set of training examples. The SVM separates these examples by a hyperplane defined by a weight vector $\vec{w}$ and a threshold b, see Figure 1. The weight vector $\vec{w}$ determines a direction perpendicular to the hyperplane, while b determines the distance to the hyperplane from the origin. A new example $\vec{z}$ is classified according to which side of the hyperplane it belongs to. From the solution of the optimization problem, the weight vector $\vec{w}$ has an expansion in a subset of the training examples, so classifying a new example $\vec{z}$ is:

$$f(\vec{z}) = \text{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i K(\vec{x}_i, \vec{z}) + b\right) \qquad (1)$$

where the α_i variables are determined by the optimization procedure and $K(\vec{x}_i, \vec{z})$ is the inner product between the example vectors.

The examples marked with grey circles in Figure 1 are called Support Vectors. These examples uniquely define the hyperplane, so if the algorithm is re-trained using only the support vectors as training examples, the same separating hyperplane is found. When examples are not linearly separable, the SVM algorithm allows for the use of slack variables for allowing classification errors and the possibility to map examples to a (high-dimensional) feature space. In this feature space, a separating hyperplane can be found such that, when mapped back to input space, describes a non-linear decision

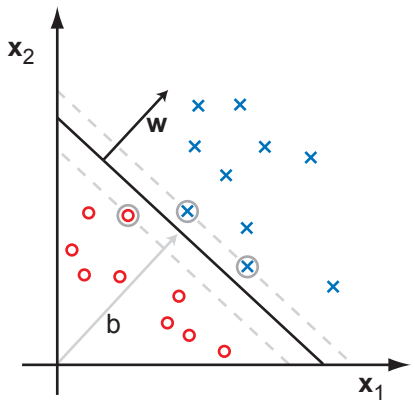

Figure 1: A maximum margin hyperplane separating a set of examples in R^2. Support Vectors are marked with circles.

function. The implicit mapping is performed by a *kernel* function that expresses the inner product between two examples in the desired feature space. This function replaces the function $K(\vec{x}_i, \vec{z})$ in Equation 1.

In our experiments, we use three standard kernel functions — the basic linear kernel, the polynomial kernel, and the radial basis kernel:[8]

- Linear: $K(\vec{x}_i, \vec{z}) = \vec{x}_i \cdot \vec{z}$
- Polynomial: $K(\vec{x}_i, \vec{z}) = (\vec{x}_i \cdot \vec{z})^d$
- Radial Basis: $K(\vec{x}_i, \vec{z} = exp(-\gamma \|\vec{x}_i - \vec{z}\|^2)$

For all experiments, we select $d = 3$ for the polynomial kernel and $\gamma = 1.0$ for the radial basis kernel. These parameters are selected as default values and are not optimized.

5 Experiments and Results

In these experiments, we use a one-against-all learning method, which means that we train one classifier for each category (and representation). When using the classifiers to predict the class of a test example, there are four possible outcomes; true positive (TP), true negative (TN), false positive (FP), and false negative (FN). Positive means that the document was classified as belonging to the category, negative that it was not, whereas true means that the classification was correct and false that it was not. From these four outcomes, we can define the standard evaluation metrics precision $\mathcal{P} = TP/(TP + FP)$ and recall $\mathcal{R} =$

[8] We use a modified version of SVM^{light} that is available at: http://svmlight.joachims.org/

$TP/(TP+FN)$. We report our results as a combined score of precision and recall, the micro-averaged $\mathcal{F}_1$ score:[9]

$$\mathcal{F}_1 = \frac{2 * \mathcal{P} * \mathcal{R}}{\mathcal{P} + \mathcal{R}}$$

There are a number of parameters that need to be optimized in this kind of experiment, including the weighting scheme, the kernel function, and the dimensionality of the BoC vectors. For ease of exposition, we report the results of each parameter set separately. Since we do not experiment with feature selection in this investigation, our results will be somewhat lower than other published results that use SVM with optimized feature selection. Our main focus is to compare results produced with BoW and BoC representations, and not to produce a top score for the Reuters-21578 collection.

5.1 Weighting Scheme

Using appropriate word weighting functions is known to improve the performance of text categorization (Yang and Pedersen, 1997). In order to investigate the impact of using different word weighting schemes for concept-based representations, we compare the performance of the SVM using the following three weighting schemes: *tf*, *idf*, and $tf \times idf$.

The results are summarized in Table 1. The BoW run uses the linear kernel, while the BoC runs use the polynomial kernel. The numbers in boldface are the best BoC runs for *tf*, *idf*, and $tf \times idf$, respectively.

	tf	*idf*	$tf \times idf$
BoW	82.52	80.13	82.77
BoC 500-dim	79.97	80.18	81.25
BoC 1,000-dim	80.31	80.87	81.93
BoC 1,500-dim	80.41	80.81	81.79
BoC 2,000-dim	80.54	80.85	82.04
BoC 2,500-dim	80.64	**81.19**	82.18
BoC 3,000-dim	80.67	81.15	82.11
BoC 4,000-dim	80.60	81.07	82.24
BoC 5,000-dim	**80.78**	81.09	**82.29**
BoC 6,000-dim	**80.78**	81.08	82.12

Table 1: Micro-averaged $\mathcal{F}_1$ score for *tf*, *idf* and $tf \times idf$ using BoW and BoC representations.

[9]Micro-averaging means that we sum the TP, TN, FP and FN over all categories and then compute the $\mathcal{F}_1$ score. In macro-averaging, the $\mathcal{F}_1$ score is computed for each category, and then averaged.

As expected, the best results for both BoW and BoC representations were produced using $tf \times idf$. For the BoW vectors, *tf* consistently produced better results than *idf*, and it was even better than $tf \times idf$ using the polynomial and radial basis kernels. For the BoC vectors, the only consistent difference between *tf* and *idf* is found using the polynomial kernel, where *idf* outperforms *tf*.[10] It is also interesting to note that for *idf* weighting, all BoC runs outperform BoW.

5.2 Parameterizing RI

In theory, the quality of the context vectors produced with the Random Indexing process should increase with their dimensionality. Kaski (1999) show that the higher the dimensionality of the vectors, the closer the matrix RR^T will approximate the identity matrix, and Bingham and Mannila (2001) observe that the mean squared difference between RR^T and the identity matrix is about $\frac{1}{k}$, where k is the dimensionality of the vectors. In order to evaluate the effects of dimensionality in this application, we compare the performance of the SVM with BoC representations using 9 different dimensionalities of the vectors. The index vectors consist of 4 to 60 non-zero elements ($\approx 1\%$ non-zeros), depending on their dimensionality. The results for all three kernels using $tf \times idf$-weighting are displayed in Figure 2.

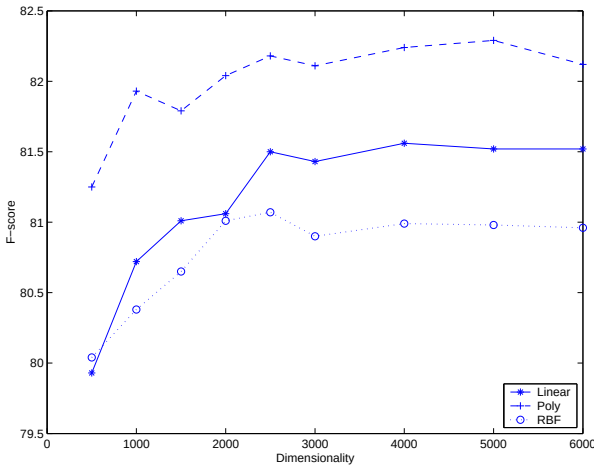

Figure 2: Micro-averaged $\mathcal{F}_1$ score for three kernels using 9 dimensionalities of the BoC vectors.

Figure 2 demonstrates that the quality of the concept-based representations increase with their dimensionality as expected, but that the

[10]For the linear and radial basis kernels, the tendency is that *tf* in most cases is better than *idf*.

increase levels out when the dimensionality becomes sufficiently large; there is hardly any difference in performance when the dimensionality of the vectors exceeds 2,500. There is even a slight tendency that the performance decreases when the dimensionality exceeds 5,000 dimensions; the best result is produced using 5,000-dimensional vectors with 50 non-zero elements in the index vectors.

There is a decrease in performance when the dimensionality of the vectors drops below 2,000. Still, the difference in $\mathcal{F}_1$ score between using 500 and 5,000 dimensions with the polynomial kernel and $tf \times idf$ is only 1.04, which indicates that Random Indexing is very robust in comparison to, e.g., singular value decomposition, where choosing appropriate dimensionality is critical.

5.3 Parameterizing SVM

Regarding the different kernel functions, Figure 2 clearly shows that the polynomial kernel produces consistently better results for the BoC vectors than the other kernels, and that the linear kernel consistently produces better results than the radial basis kernel. This could be a demonstration of the difficulties of parameter selection, especially for the γ parameter in the radial basis kernel. To further improve the results, we can find better values of γ for the radial basis kernel and of d for the polynomial kernel by explicit parameter search.

6 Comparing BoW and BoC

If we compare the best BoW run (using the linear kernel and $tf \times idf$-weighting) and the best BoC run (using 5,000-dimensional vectors with the polynomial kernel and $tf \times idf$-weighting), we can see that the BoW representations barely outperform BoC: 82.77% versus 82.29%. However, if we only look at the results for the ten largest categories in the Reuters-21578 collection, the situation is reversed and the BoC representations outperform BoW. The $\mathcal{F}_1$ measure for the best BoC vectors for the ten largest categories is 88.74% compared to 88.09% for the best BoW vectors. This suggests that BoC representations are more appropriate for large-size categories.

The best BoC representations outperform the best BoW representations in 16 categories, and are equal in 6. Of the 16 categories where the best BoC outperform the best BoW, 9 are better only in recall, 5 are better in both recall and precision, while only 2 are better only in precision.

It is always the same set of 22 categories where the BoC representations score better than, or equal to, BoW.[11] These include the two largest categories in Reuters-21578, *"earn"* and *"acq"*, consisting of 2,877 and 1,650 documents, respectively. For these two categories, BoC representations outperform BoW with 95.57% versus 95.36%, and 91.07% versus 90.16%, respectively. The smallest of the "BoC categories" is *"fuel"*, which consists of 13 documents, and for which BoC outperforms BoW representations with 33.33% versus 30.77%. The largest performance difference for the "BoC categories" is for category *"bop"*, where BoC reaches 66.67%, while BoW only reaches 54.17%. We also note that it is the same set of categories that is problematic for both types of representations; where BoW score 0.0%, so does BoC.

7 Combining Representations

The above comparison suggests that we can improve the performance of the SVM by combining the two types of representation. The best $\mathcal{F}_1$ score can be achieved by selecting the quadruple (TP, FP, TN, FN) for each individual category from either BoW or BoC so that it maximizes the overall score. There are 2^{90} such combinations, but by expressing the $\mathcal{F}_1$ function in its equivalent form $\mathcal{F}_1 = (2 * TP)/(2 * TP + FP + FN)$, we can determine that for our two top runs there are only 17 categories such that we need to perform an exhaustive search to find the best combination. For instance, if for one category both runs have the same TP but one of the runs have higher FP and FN, the other run is selected for that category and we do not include that category in the exhaustive search.

Combining the best BoW and BoC runs increases the results from 82.77% (the best BoW run) to **83.91**%. For the top ten categories, this increases the score from 88.74% (the best BoC run) to **88.99**%. Even though the difference is admittedly small, the increase in performance when combining representations is not negligible, and is consistent with the findings of previous research (Cai and Hofmann, 2003).

[11] The "BoC categories" are: *veg-oil, heat, gold, soybean, housing, jobs, nat-gas, cocoa, wheat, rapeseed, livestock, ship, fuel, trade, sugar, cpi, bop, lei, acq, crude, earn, money-fx.*

8 Conclusions

We have introduced a new method for producing concept-based (BoC) text representations, and we have compared the performance of an SVM classifier on the Reuters-21578 collection using both traditional word-based (BoW), and concept-based representations. The results show that BoC representations outperform BoW when only counting the ten largest categories, and that a combination of BoW and BoC representations improve the performance of the SVM over all categories.

We conclude that concept-based representations constitute a viable supplement to word-based ones, and that there are categories in the Reuters-21578 collection that benefit from using concept-based representations.

9 Acknowledgements

This work has been funded by the European Commission under contract IST-2000-29452 (DUMAS — Dynamic Universal Mobility for Adaptive Speech Interfaces).

References

R. Baeza-Yates and B. Ribeiro-Neto. 1999. *Modern Information Retrieval*. ACM Press / Addison-Wesley.

D. Baker and A. McCallum. 1998. Distributional clustering of words for text classification. In *SIGIR 1998*, pages 96–103.

Ella Bingham and Heikki Mannila. 2001. Random projection in dimensionality reduction: applications to image and text data. In *Knowledge Discovery and Data Mining*, pages 245–250.

Lijuan Cai and Thomas Hofmann. 2003. Text categorization by boosting automatically extracted concepts. In *SIGIR 2003*, pages 182–189.

S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41(6):391–407.

S. Dumais, J. Platt, D. Heckerman, and M. Sahami. 1998. Inductive learning algorithms and representations for text categorization. In *Proceedings of ACM-CIKM98*, pages 148–155.

R. Hecht-Nielsen. 1994. Context vectors: general purpose approximate meaning representations self-organized from raw data. In J.M. Zurada, R.J. Marks II, and C.J. Robinson, editors, *Computational Intelligence: Imitating Life*, pages 43–56. IEEE Press.

W.B. Johnson and J. Lindenstrauss. 1984. Extensions of lipshitz mapping into hilbert space. *Contemporary Mathematics*, 26:189–206.

P. Kanerva, J. Kristofersson, and A. Holst. 2000. Random indexing of text samples for latent semantic analysis. In *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, page 1036. Erlbaum.

J. Karlgren and M. Sahlgren. 2001. From words to understanding. In Y. Uesaka, P. Kanerva, and H. Asoh, editors, *Foundations of Real-World Intelligence*, pages 294–308. CSLI Publications.

S. Kaski. 1999. Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *Proceedings of the IJCNN'98, International Joint Conference on Neural Networks*, pages 413–418. IEEE Service Center.

T. Landauer and S. Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240.

D. Lewis. 1992. An evaluation of phrasal and clustered representations on a text categorization task. In *SIGIR 1992*, pages 37–50.

K. Lund, C. Burgess, and R. A. Atchley. 1995. Semantic and associative priming in high-dimensional semantic space. In *Proceedings of the 17th Annual Conference of the Cognitive Science Society*, pages 660–665. Erlbaum.

C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. 1998. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the 17th ACM Symposium on the Principles of Database Systems*, pages 159–168. ACM Press.

F. Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

Y. Yang and J. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420.