

An Empirical Comparison of Text Categorization Methods

Ana Cardoso-Cachopo^{1,2} and Arlindo Limede Oliveira^{1,2}

¹ Instituto Superior Técnico
Departamento de Engenharia Informática
Av. Rovisco Pais, 1
1049-001 Lisboa — Portugal
² INESC-ID
Rua Alves Redol, 9
1000-029 Lisboa — Portugal
acardoso@gia.ist.utl.pt, aml@inesc.pt

Abstract. In this paper we present a comprehensive comparison of the performance of a number of *text categorization* methods in two different data sets. In particular, we evaluate the Vector and Latent Semantic Analysis (LSA) methods, a classifier based on Support Vector Machines (SVM) and the k-Nearest Neighbor variations of the Vector and LSA models.

We report the results obtained using the Mean Reciprocal Rank as a measure of overall performance, a commonly used evaluation measure for *question answering* tasks. We argue that this evaluation measure is also very well suited for *text categorization* tasks.

Our results show that overall, SVMs and k-NN LSA perform better than the other methods, in a statistically significant way.

1 Introduction

As the amount of information in written form increases, *text categorization* techniques become more necessary to find relevant information in a variety of tasks such as finding answers to similar questions, classifying news by subject or news-group, sorting e-mail messages, etc.

A number of approaches to *text categorization* has been proposed. The goal of *text categorization* methods is to associate one (or more) of a given set of categories to a particular document. They differ on how they represent documents and on how they decide which category to assign to a particular document. As such, a particular approach can be more suitable for a particular task, with a specific dataset, while another one is better adapted to a different setting. The measure used to quantify the performance of each approach can also influence the results.

In this paper we compare how the Vector Model, Latent Semantic Analysis (LSA), a classifier based on Support Vector Machines (SVM) and the k-Nearest Neighbors variations of the Vector and LSA models perform on two different data sets, using the Mean Reciprocal Rank as a measure of overall performance.

2 Data Description

As each model can behave differently in different settings, we have to test them in the same setting if we want to be able to compare the results. To allow for the generalization of results, we should use more than one test problem. For our study we used two different sets of data, in two different languages.

Financial Institution’s Dataset The first dataset consists of messages sent by the clients of a financial institution to their help-desk and the answers they got from the help desk assistants. All these documents are in Portuguese. We had access to the collection of answered messages under a non-disclosure agreement.

We classified the data according to the type of request that was made. This dataset contains one class for each type of message that can be answered automatically and one class that comprises all the messages that need human intervention. Our complete dataset has 1391 classified messages and the respective answers. There are 37 different classes, containing from 5 to 346 messages each.

For the purpose of model comparison 10 classes were selected, containing 34 to 58 messages each, in a total of 461 messages. We will call this dataset **C10**. Table 1 lists the message types and the number of messages per type³.

Type of message	# msgs
Incomplete data for a request of credit allowance	34
Question about the available amount of credit	36
General information about “points promotion”	42
More information on “low value payment” operation	45
Request for the receipt of an operation	46
Impossible to request checks through the internet	46
Points in the “points promotion” have disappeared	46
Impossible to buy or sell stock funds through the internet	52
How to buy or sell stocks through the internet	56
What is the price of a particular financial operation?	58
Total	461

Table 1. Number of messages of each type in C10

In this study, we will describe results that use only the client’s message for training.

20 Newsgroups Dataset The second dataset is a subset of the 20 Newsgroups dataset, that can be downloaded from UCI’s Knowledge Discovery in Databases Archive. Here, one thousand Usenet articles were taken from each of the 20 newsgroups in Table 2. Approximately 4% of the articles are cross-posted. The articles are typical postings and thus have headers including subject lines, signature files, and quoted portions of other articles.

³ Some of these operations make sense only in Portugal’s banking systems.

Group	Group
alt.atheism	rec.sport.hockey
comp.graphics	sci.crypt
comp.os.ms-windows.misc	sci.electronics
comp.sys.ibm.pc.hardware	sci.med
comp.sys.mac.hardware	sci.space
comp.windows.x	soc.religion.christian
misc.forsale	talk.politics.guns
rec.autos	talk.politics.mideast
rec.motorcycles	talk.politics.misc
rec.sport.baseball	talk.religion.misc

Table 2. Usenet groups for the 20 Newsgroups dataset

We used a subset of the 20 Newsgroups Dataset, containing a total of 100 messages from each newsgroup. Here, 18 messages were cross-posted. We will call this dataset **mini20**.

2.1 Pre-processing the Data

It is widely accepted that the way that documents and queries are represented influences the quality of the results that can be achieved. With this fact in mind, there are several proposals that aim at improving retrieval results. However, it is seldom guaranteed that they achieve their goal.

The main aim of preprocessing the data is to reduce the problem’s dimensionality by controlling the size of the system’s vocabulary (different index terms). In some situations, aside from reducing the complexity of the problem, this preprocessing will also unify the data in a way that improves performance.

In this work, we applied some of the filters used routinely in Information Retrieval:

- Discard words shorter than 3 or longer than 20 characters.
- Remove numbers and non-letter characters.
- Case and special character unification. Special character unification is needed in Portuguese, because there are accentuated characters and some people use them and some don’t.

3 Applied Methods

In this work we are concerned with models for the categorization of natural language text. That is, models that, given a set of training documents with known categories and a new document, which is usually called the *query*, will predict the query’s category.

In these models, usually based on statistical analysis, a text document is represented as a set of index terms or keywords. Each index term corresponds to a word in the initial text and has a weight associated to it, which should reflect

how important this index term is, for that document and/or for the collection of documents.

For efficiency reasons, some of these models make simplifications that may not be totally justified but that have been experimentally validated. Some of these simplifications are:

1. They ignore the natural language structure of text. They do not try to fully “understand” a document, but they can use the structure that is easy to find (like HTML tags, for instance), even when they are processing large amounts of information. Besides being more efficient, this approach also has the advantage of not using domain-dependent techniques.
2. They assume that weights for the index terms are mutually independent. Although this simplification allows for a much easier treatment of the documents, weights for the index terms usually are not independent, because the fact that one of the index terms appears in the text may increase the probability of finding another term that is usually related to it, as in “computer network”, for instance.
3. They ignore the order of words. This way, all the texts that are permutations of the same words are considered equal. This simplification is not always justified, but it is necessary for efficiency reasons.

In the next sections we briefly describe the methods that we compare in this paper.

Baeza-Yates and Ribeiro-Neto [1] includes a description of most Information Retrieval models and Sebastiani [19] contributed a more up-to-date survey of machine learning methods used for automated Text Categorization.

3.1 Vector Model

In the Vector Model [18, 16], documents are represented as a set of index terms which are weighted according to their importance for a particular document and for the general collection.

The index terms usually correspond to the words or tokens in the document (or query) and index term weights can be computed in several ways. The most usual is tf-idf (term frequency – inverse document frequency) [17], which increases with the number of times that the term occurs in the document and decreases with the number of times the term occurs in the collection. The weight of term t_i for document d_j is

$$w_{i,j} = \frac{freq_{i,j}}{\max_l(freq_{l,j})} \times \log \frac{N}{n_i}$$

where $freq_{i,j}$ is the raw frequency of term t_i on document d_j , N the total number of documents and n_i the number of documents where term t_i appears.

For the weights of the index terms in the query, Salton and Buckley [17] suggest

$$w_{i,j} = \left(0.5 + \frac{0.5 freq_{i,j}}{\max_l(freq_{l,j})} \right) \times \log \frac{N}{n_i}$$

Documents and queries are then represented as vectors in an M -dimensional space, where M is the total number of index terms.

Based on the weights of its terms, documents can be ranked by a decreasing order of similarity to the query. The similarity of each document d_j to the query q is computed as the cosine of the angle formed by the vectors that represent each of them

$$\text{sim}(d_j, q) = \frac{\vec{d}_j \cdot \vec{q}}{\|\vec{d}_j\| \times \|\vec{q}\|}$$

The category of the query can be determined as the category of the most similar document found.

3.2 Latent Semantic Analysis/Indexing Model

Matching documents and queries solely based on index terms can be misleading, because a document can be relevant for a query without having any terms in common with it.

The idea behind the Latent Semantic Analysis (or Latent Semantic Indexing) model (LSA/I) [11, 10] is to map each document and query vector into a lower dimensional space which is associated with concepts and retrieve the documents in this space. Arguably, retrieval effectiveness in this space will be better and it will also be computationally less costly, because it is a lower dimensional space.

LSA starts with a term-by-document rectangular matrix X which is decomposed by singular value decomposition (SVD) into the product of three other matrices: $X = T_0 S_0 D_0$, such that T_0 and D_0 have orthonormal columns and S_0 is diagonal. T_0 and D_0 are the matrices of *left* and *right singular vectors* and S_0 is the diagonal matrix of *singular values*. If the singular values in S_0 are ordered by size (and the corresponding row and column permutations applied to T_0 and D_0), the first largest k may be kept and the remaining ones set to zero. The product of the resulting matrices is a matrix \hat{X} which is only approximately equal to X and is of rank k . It can be shown that the new matrix \hat{X} is the matrix of rank k which is closest in the least squares sense to X . Ideally, we want a value of k that is large enough to fit all the real structure in the data, but small enough so that we do not overfit the data.

After these transformations the result can still be represented geometrically by a spatial configuration in which the cosine between vectors representing a document and a query corresponds to their similarity.

As in the Vector Model, documents can now be ranked according to their similarity to the query, and the category of the query is the category of the most similar document.

3.3 Support Vector Machines

The Support Vector Machines model or large margin classifier was introduced by Vapnik [20, 7] and was first applied for *text categorization* by Joachims [12, 13]. A thorough description can also be found in other sources [9, 3].

Support Vector Machines (SVMs) is a method for efficiently training linear classifiers. This technique is based on recent advances in statistical learning theory. They map the documents into a high dimensional feature space, and try to learn a separating hyperplane, that provides the widest margins between two different types of documents. SVMs use Lagrange multipliers to translate the problem of finding this hyperplane into an equivalent quadratic optimization problem for which efficient algorithms exist, and which are guaranteed to find the global optimum. The set of coefficients α_i^* resulting from the optimization process can then be used to construct the hyperplane that correctly classifies all the training examples with the maximum margin:

$$\vec{w} \cdot \vec{d} = \sum_{i=1}^n \alpha_i^* y_i (\vec{d}_i \cdot \vec{d}) \quad \text{and} \quad b = \frac{1}{2}(\vec{w} \cdot \vec{d}_\circ + \vec{w} \cdot \vec{d}_\bullet)$$

This equation shows that the resulting weight vector of the hyperplane is constructed as a linear combination of the training examples. Only examples for which the coefficient α_i^* is greater than zero contribute. These are called the *support vectors*, because they have minimum distance to the hyperplane. Figure 1 illustrates these ideas.

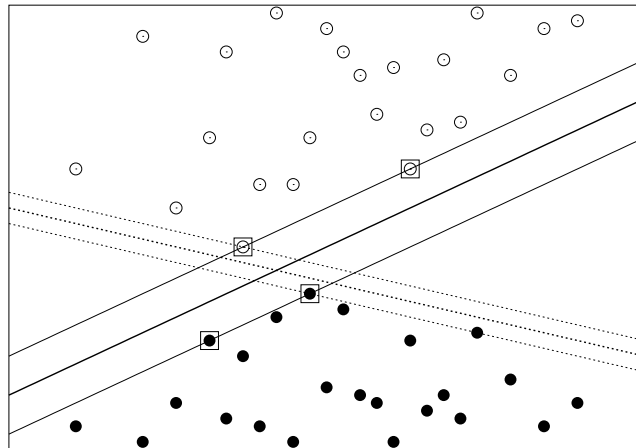


Fig. 1. Example of a two class, linearly separable problem and two possible separation hyperplanes with corresponding margins. The decision hyperplane chosen by SVMs is the bold solid line, which corresponds to the largest possible separation margins. The squares indicate the corresponding support vectors.

For sets of documents that are not linearly separable, SVMs use *convolution functions* (or kernels), that transform the initial feature space into another one, where they are able to find the hyperplane that separates the data with the widest margin.

By including several classifiers, these ideas can easily be generalized for datasets with more than two classes of documents.

3.4 k-Nearest Neighbors

The initial application of k-NN to *text categorization* was reported by Masand and colleagues [8, 15]. The idea is to determine the category of a given query based on the categories of the k documents that are nearest to it in the document space.

For this study, we first calculated each document’s similarity to the query. Then, we used a voting strategy to find the query’s class: each retrieved document contributes a vote for its class, weighted by its similarity to the query. The query’s possible classifications will be ranked according to the votes they got in the previous step.

In this paper, we used a vector-based, distance-weighted matching function, as did Yang [22, 23], by calculating the distance using the Vector Model. We refer to it by “k-NN Vector”. We also calculated the distance using the similarity obtained with LSA and called it “k-NN LSA”. Because the number of examples of each class in C10 and mini20 is approximately the same, we used all the examples in each dataset as k .

4 Evaluation Measure

Evaluating the performance of computational systems is often done in terms of the resources (time and space) they need to operate, assuming that they perform the task that they are supposed to.

However, in Information Retrieval, it is not enough to retrieve a set of documents in a reasonable amount of time. The retrieved documents should also be the “right” ones, that is, the ones that are relevant for the query. Precision and recall are two measures that have been widely used to compare the performance of Information Retrieval models. *Precision* is defined as the fraction of the retrieved documents that are relevant, that is $\frac{RRD}{RetD}$ (it can be viewed as a measure of the system’s soundness). *Recall* is defined as the fraction of the relevant documents that is actually retrieved, that is $\frac{RRD}{RelD}$ (it can be viewed as a measure of the system’s completeness). Here, RRD means the number of retrieved documents that are actually relevant, $RetD$ means the total number of retrieved documents and $RelD$ means the total number of relevant documents.

However, to evaluate *question answering* or *answer finding* systems, the measures used to evaluate general IR systems are not very adequate [2]. We want to get the right answer, we want it only once (recall is not very important), and we want it as close to the first answer as possible. So we will be interested in a measure that takes the rank of the first correct answer into account.

The Mean Reciprocal Rank (MRR), is a measure used to evaluate submissions to the “Question Answering Track” of the TREC conferences, defined by Voorhees [21]. The idea is that its value is higher if the rank of the first correct

answer is lower. We think that this measure is more adequate than the ones based on the values of precision and recall, which are more adequate for general Information Retrieval tasks. The MRR also has several advantages:

1. It is closely related to the average precision measure used extensively in document retrieval.
2. It is bounded between 0 (worst) and 1 (best), inclusive, and averages well.
3. A run is penalized for not retrieving any correct answer for a question, but not unduly so.
4. It is intuitive and easy to calculate.

The MRR can be calculated for each individual question as the reciprocal of the rank at which the first correct response was returned, or 0 if none of the first N responses contained a correct answer. The score for a sequence of questions is the mean of the individual question's reciprocal ranks.

However, the MRR also has some drawbacks:

1. The score for an individual question can take only $N + 1$ values.
2. *Question answering* systems are given no credit for retrieving multiple (different) correct answers.
3. Since the track required at least one response for each question, a system could receive no credit for realizing it did not know the answer.

In *text categorization*, we are also interested in getting the right category for a document (assuming each document has only one category), we need it only once, and the closer to the top position the better, because it gives us a measure of the system's confidence in the answer. Therefore, we decided to report the results obtained using the Mean Reciprocal Rank as a measure of overall performance. The Expected Search Length [6] was a good alternative, but we decided to use MRR because it is a common evaluation measure used for *question answering* tasks.

In our work, we considered the top ten categories returned by each model to calculate the MRR.

5 Experimental Setup

In our experiments we used existing implementations for each model. These implementations are freely available, and can be obtained from the authors.

For the Vector Model we used a Sourceforge project called IGLU <http://sourceforge.net/projects/iglu-java>. IGLU aims at being a software platform suitable for testing Information Retrieval models. At the time of this writing, only the Vector Model is implemented.

For LSA we used FAQO — Frequently Asked Questions Organizer [4]. FAQO is an application that was designed to help the technical support team at Unidata (University Corporation for Atmospheric Research) in the task of answering questions posed by the users of their software. It uses LSA/I to find similarities between the user's questions and questions that were previously answered by

Unidata’s personnel. As a result, FAQO shows a ranked list of previous questions and answers that are most similar to the present one. FAQO is an open source project released under the GPL license.

For SVMs we used a library called LIBSVM [5]. LIBSVM is an integrated software for Support Vector classification (among others) that supports multi-class classification. Their goal is to help users from other fields to easily use SVM as a tool. LIBSVM provides a simple interface that users can use to easily link it with their own programs.

LIBSVM already supports multi-class classification, returning, for each document, the (one) class it belongs to. However, to calculate the MRR, we need a ranked list of possible classes for each document, as is returned by the other models we are using. So, to determine the class of a given document, we implemented a “voting strategy”, where a document’s possible classes are ranked according to the number of votes that they had in a one-against-one approach, as Chang and Lin did [5].

We also used this “voting strategy”, now weighted according to the similarity measure returned by the Vector or LSA models, to implement the k-NN variations of these models.

6 Results

To see if the number of terms used in the experiments influences the models’ performance, the terms are ordered according to the information gain criterion [24, 12] and only the first most informative ones are selected. We performed extensive testing to determine the ideal number of terms that should be considered by each model.

For the datasets C10 and mini20, respectively, the lines in Figures 2 and 3 represent the mean of the MRR for a five-fold cross-validation test for each of the five models under evaluation, depending on the number of terms that is used.

For C10, which has a total of 2179 terms, we can see the whole range for the number of terms used in the tests. However, for mini20, which has a total of 33444 terms, we shortened the range for the number of terms, because we observed that the variation of the MRR was not interesting beyond this value. The “dots” to the right of the graphs show the value of the MRR using *all* the existing terms.

As we can observe by comparing both figures, the lines have approximately the same shape, which suggests that these results are not data-dependent. They should not depend on the language either, because each dataset is in a different language.

If we compare the performance of the Vector model with the performance of k-NN Vector, we observe that they do not show very significant differences. While the Vector Model is worse than the k-NN Vector for a smaller number of terms for C10, and improves over k-NN Vector as the number of terms increases, the opposite is true for mini20.

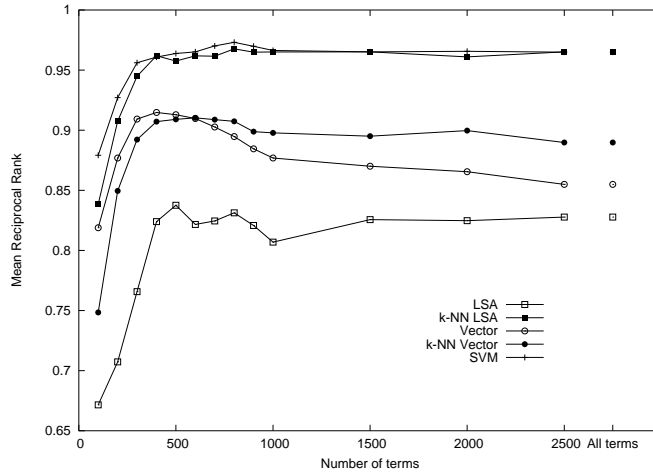


Fig. 2. MRR values for C10

However, if we compare the performance of LSA with its k-NN variation, we see that k-NN LSA behaves significantly better, and that this happens independently of the number of terms that we choose for both datasets.

While LSA is the worst performing method for C10, it performs significantly better than the Vector model and its k-NN variation for mini20. This shows that LSA performs better when it has many training examples.

The only model that has a performance comparable with k-NN LSA for C10 is the SVM-based, which even outperforms k-NN LSA for some values of numbers of features used. For mini20, SVMs are consistently above LSA.

We can also observe that, for all models, the results obtained using only a limited number of the most informative terms are not much worse than the results using all the terms in the dataset. On the contrary, in the case of the C10 dataset, the best results are obtained when we use between 400 and 800 terms. Reducing the number of terms used results in a considerable reduction in computational effort. If this is not of utmost importance for C10, it can lead to a considerable computational gain for mini20.

7 Statistical Significance Tests

To confirm if the differences observed between each pair of models are statistically significant, we performed paired t-tests for each pair of models. For each model, we chose the number of terms that provided the higher average MRR.

The values obtained for the MRR for each model, in each of the 5-folds used for cross-validation and their respective mean, are presented in Table 3 for the C10 dataset and in Table 4 for the mini20 dataset. The results of the paired t-tests between each possible pair of models are depicted in Tables 5 and 6, respectively.

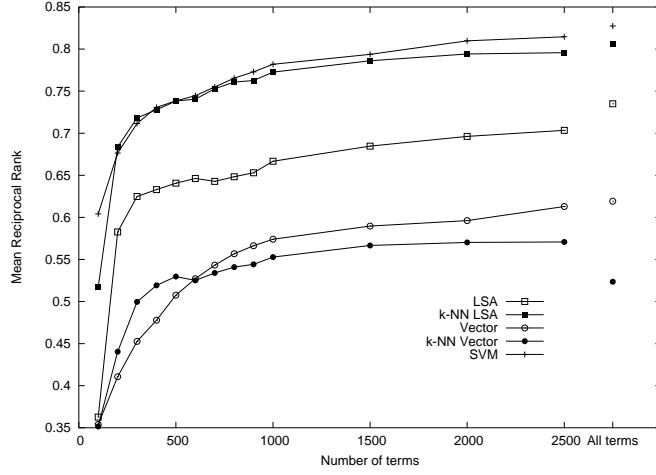


Fig. 3. MRR values for mini20 using at most 2500 terms

C10

	Vector	k-NN Vector	LSA	k-NN LSA	SVM, $\gamma = 0.4$
# terms	400	600	500	800	800
fold 1	0.9355	0.8922	0.8545	0.9764	0.9810
fold 2	0.8905	0.9101	0.8288	0.9602	0.9764
fold 3	0.9067	0.9184	0.8641	0.9656	0.9683
fold 4	0.9592	0.9402	0.8250	0.9837	0.9946
fold 5	0.8820	0.8907	0.8155	0.9525	0.9453
mean	0.9148	0.9103	0.8376	0.9677	0.9731

Table 3. MRR for 5-fold cross-validation for C10.

In these tables, $A \sim B$ means that the results obtained with model A are not statistically different from those obtained with model B , and $A \gg B$ means that the results obtained with model A are statistically better than those obtained with model B . In each case we present the p value, which is the significance of each paired t-test.

From the results in Table 5, we can extract a partial order among the various models, for dataset C10, when we consider only the number of terms that provide the higher MRR:

$$\{\text{k-NN LSA, SVM}\} \gg \{\text{Vector, k-NN Vector}\} \gg \text{LSA}$$

From the results in Table 6, we can extract a total order among the various models, for dataset mini20, when we consider only the number of terms that provide the higher MRR:

$$\text{SVM} \gg \text{k-NN LSA} \gg \text{LSA} \gg \text{Vector} \gg \text{k-NN Vector}$$

mini20					
	Vector	k-NN Vector	LSA	k-NN LSA	SVM, $\gamma = 0.4$
# terms	18000	2500	all	12000	all
fold 1	0.6240	0.5672	0.7366	0.8040	0.8203
fold 2	0.6269	0.5341	0.7357	0.8096	0.8254
fold 3	0.6346	0.6246	0.7276	0.8198	0.8318
fold 4	0.6147	0.5571	0.7098	0.7950	0.8196
fold 5	0.6670	0.5708	0.7651	0.8274	0.8399
mean	0.6334	0.5707	0.7350	0.8112	0.8274

Table 4. MRR for 5-fold cross-validation for mini20.

c10			
Model 1	t-test	Model 2	p value
Vector	~	k-NN Vector	0.7221
Vector	≫	LSA	0.0038
k-NN LSA	≫	Vector	0.0020
SVM	≫	Vector	0.0012
k-NN Vector	≫	LSA	0.0026
k-NN LSA	≫	k-NN Vector	0.0007
SVM	≫	k-NN Vector	0.0004
k-NN LSA	≫	LSA	0.0001
SVM	≫	LSA	0.0001
k-NN LSA	~	SVM	0.1581

Table 5. Results of the t-test for dataset C10.

8 Conclusions

In this paper we presented experiments using k-NN LSA, a new combination of the standard k-NN method on top of LSA that performed almost as well as the best performing methods for *text categorization* reported so far. To the best of our knowledge, there are no published results using the k-NN LSA.

We used the MRR, an evaluation measure that is very adequate for one-class *text categorization* tasks.

We showed that overall, SVMs and k-NN LSA perform better than the other methods, in a statistically significant way. As future work, we plan to investigate if the results obtained in this set of two text classification problems can be replicated in other benchmarks. We also plan to investigate if further improvements can be applied to the SVMs and k-NN LSA models. If possible, this would further enhance the superiority of these methods observed in this experiments.

mini20

Model 1	t-test	Model 2	p value
Vector	≫	k-NN Vector	0.0079
LSA	≫	Vector	0.0000
k-NN LSA	≫	Vector	0.0000
SVM	≫	Vector	0.0000
LSA	≫	k-NN Vector	0.0004
k-NN LSA	≫	k-NN Vector	0.0000
SVM	≫	k-NN Vector	0.0000
k-NN LSA	≫	LSA	0.0001
SVM	≫	LSA	0.0001
SVM	≫	k-NN LSA	0.0010

Table 6. Results of the t-test for dataset mini20.

References

1. Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, Reading, Massachusetts, USA, 1999.
2. Adam Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu O. Mittal. Bridging the lexical chasm: statistical approaches to answer-finding. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 192–199, Athens, Greece, July 2000.
3. Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
4. John Caron. Experiments with LSA scoring: Optimal rank and basis. In *Proceedings of SIAM Computational Information Retrieval Workshop*, Raleigh, NC, USA, October 2000.
5. Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
6. William S. Cooper. Expected search length: a single measure of retrieval effectiveness based on weak ordering action of retrieval systems. *Journal of the American Society for Information Science*, 19(1):30–41, 1968.
7. Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, November 1995.
8. Robert M. Creedy, Brij M. Masand, Stephen J. Smith, and David L. Waltz. Trading MIPS and memory for knowledge engineering: classifying census returns on the Connection Machine. *Communications of the ACM*, 39(1):48–63, January 1996.
9. Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, MA, USA, 2000.
10. Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.
11. George W. Furnas, Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, Richard A. Harshman, L.A. Streeter, and K.E. Lochbaum. Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 465–480, Grassau, France, June 1988.

12. Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, Germany, 1998. Springer-Verlag. Published in the “Lecture Notes in Computer Science” series, number 1398.
13. Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of the 16th International Conference on Machine Learning*, pages 200–209, Bled, Slovenia, 1999. Morgan Kaufmann Publishers, Inc.
14. Karen Spark Jones and Peter Willett, editors. *Readings in Information Retrieval*. Morgan Kaufmann Publishers, Inc., Los Altos, USA, 1997.
15. Briji Masand, Gordon Linoff, and David Waltz. Classifying news stories using memory-based reasoning. In Nicholas J. Belkin, Peter Ingwersen, and Annelise Mark Pejtersen, editors, *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 59–65, Copenhagen, Denmark, June 1992. ACM Press.
16. Gerard Salton. *The SMART Retrieval System*. Prentice-Hall, Inc., New Jersey, USA, 1971.
17. Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988. Also reprinted in [14, pages 323–328].
18. Gerard Salton and Michael Lesk. Computer evaluation of indexing and text processing. *Journal of the ACM*, 15(1):8–36, January 1968. Also reprinted in [14, pages 60–84].
19. Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
20. Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, Heidelberg, Germany, 1995.
21. Ellen M. Voorhees. The TREC-8 question answering track report. In Ellen M. Voorhees and Donna K. Harman, editors, *Proceedings of the 8th Text REtrieval Conference*, pages 77–82, Gaithersburg, Maryland, USA, November 1999.
22. Yiming Yang. Expert network: effective and efficient learning from human decisions in text categorisation and retrieval. In W. Bruce Croft and Cornelis J. van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 13–22, Dublin, Ireland, July 1994. Springer-Verlag.
23. Yiming Yang and Xin Liu. A re-examination of text categorization methods. In Marti A. Hearst, Fredric Gey, and Richard Tong, editors, *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 42–49, Berkeley, CA, USA, August 1999.
24. Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of the 14th International Conference on Machine Learning*, pages 412–420, Nashville, TN, USA, 1997. Morgan Kaufmann Publishers, Inc.