

# Efficient text categorization

Marko Grobelnik and Dunja Mladenić  
Department for Intelligent Systems, J.Stefan Institute,  
Jamova 39, 1111 Ljubljana, Slovenia  
Phone: (+38)(61) 1773 272, Fax: (+38)(61) 1258-158  
E-mail: Marko.Grobelnik@ijs.si, Dunja.Mladenic@ijs.si

February 17, 1998

## Abstract

We present an approach to text categorization using machine learning techniques. The approach is developed and tested on large text hierarchy named Yahoo that is available on the Web. We handle the large number of features and training examples by taking into account hierarchical structure of examples and using feature subset selection for large text data. The large number of categories is handled separately for each testing example by pruning unpromising categories. In this way, the number of categories to be considered is cut to less than a half without degrading the system performance. Our experiments are performed using naive Bayesian classifier on text data using feature-vector document representation that includes n-grams instead of just single words (unigrams). Experimental evaluation on three domains constructed from Yahoo hierarchy shows that among several hundred categories the correct category is assigned probability over 0.99 when rather small number of features used.

## 1 Introduction

Text categorization is gaining popularity with the growing interest and usage of text data available on World Wide Web. The problem of text categorization is well known in information retrieval community where new methods are usually tested on publicly available data bases (eg. Reuters, MEDLINE). Here we use machine learning techniques to automatically construct classifier from large text hierarchy. The existing Web hierarchy we use named Yahoo hierarchy is human constructed and captures the current situation on the Web. Our approach is not limited to Web hierarchy and can be applied on other hierarchies like for instance, thesaurus.

Text domains that we constructed from Yahoo hierarchy are described in Section 2. Usage of machine learning techniques on these domains is described in Section 3. Experimental results are given in Section 4 and discussion in Section 5.

## 2 Domain description

We developed and tested our approach using the existing Web hierarchy named Yahoo. The categories in Yahoo hierarchy are human constructed and designed for human Web browsing making the hierarchy biased toward human knowledge areas that are

represented in Web documents. For example, one of the most general categories named ‘Business and Economy’ is over-represented including more than a third of all documents, while the other category that appears at the same level of hierarchy named ‘Social Science’ includes less than 1% of all documents (see Figure 1). The Yahoo hierarchy itself is currently build on approximately 900,000 Web documents located all around the Internet. Hyperlinks to that documents are organized in about 50,000 Yahoo Web documents. Each Yahoo document represents one of the included categories. This documents are connected with hyperlinks forming a hierarchical structure with more general categories closer to the root of the hierarchy. There are currently fourteen top level Yahoo categories. Figure 1 shows them with the approximate number of Web documents each category is based on. Each of the top categories is further represented with hierarchical structure of more specific categories. As example we show in Figure 1 the part of the first-level sub-categories in ‘Science’ top category ranging from ‘Acoustics’ to ‘Weights and Measures’.

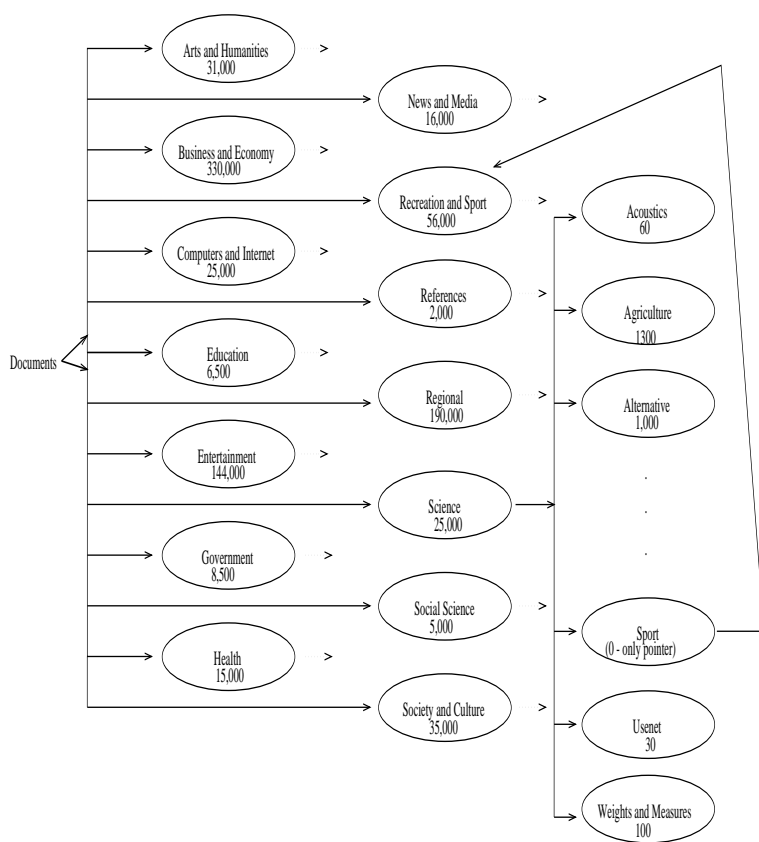


Figure 1: Top level of the Yahoo categorization and the first level of sub-categories in ‘Science’ category with the approximate number of documents in each category. Yahoo site in UK & Ireland, November 1997.

In order to apply machine learning on text data we represent documents as feature-vectors using the bag-of-words representation as commonly used in learning on text data. We also reduce the number of features by removing words contained in the “stop-list” and removing infrequent words. Our document representation includes not only single words (unigrams) but also up to 5 words (1-grams, 2-grams, ... 5-grams)

occurring in document as a sequence (eg. ‘machine learning’, ‘world wide web’). Since we have previously removed words contained in “stop-list”, some features that represent a word sequence can actually capture longer sequences like for instance, ‘Word for Windows’ that is represented as 2-gram ‘Word Windows’ or ‘winners will be posted at the end of each two-week period’ that is represented as 5-gram ‘winners posted end two-week period’. In this way we can capture some characteristic word combinations but also increase the number of feature (eg. in whole Yahoo hierarchy from 69,280 features for 1-grams to 255,602 features for 5-grams). We additionally apply feature subset selection as commonly used on text data eg. [7], [10]. In this approach a score is assigned to each feature independently, features are sorted according to the assigned score and a predefined number of the best features is taken to form the solution feature subset. As a scoring method we used Odds Ratio [8] that prefers features characteristic for positive examples.  $OddsRatio(F) = \log \frac{odds(W|pos)}{odds(W|neg)} = \log \frac{P(W|pos)(1-P(W|neg))}{(1-P(W|pos))P(W|neg)}$ . Where  $P(W|pos)$  is the conditional probability of word  $W$  occurring given the class value ‘positive’,  $P(W|neg)$  is the conditional probability of word  $W$  occurring given the class value ‘negative’. We handle singularities as proposed in [9].

The process of feature generation is performed in  $n$  passes over documents, where  $i$ -grams are generated in the  $i$ -th pass. At the end of each pass over documents all infrequent features are deleted (here we check for frequency  $< 4$ ). Each new pass generates features of length  $i$  only from the candidate features (of length  $i-1$ ) generated in the previous pass. This process is similar to the large k-itemset generation used in association rules algorithm [1].

### 3 Learning text classifier

In order to handle large number of examples and features we divide the whole problem into subproblems as suggested in [5] for hierarchical document classification. For a new document, the classifier returns probability distribution over categories included in the hierarchy. For each of the subproblems, a classifier is constructed that predicts probability that a document is a member of a corresponding category. A set of positive and negative examples for each subproblem is constructed from the given hierarchical structure. We define a set of negative examples as examples from the whole hierarchy. The set of positive examples is constructed from examples in the corresponding category node and all the examples from its subtrees, if any. For example, for Yahoo category ‘Arts: PerformingArts: Dance: Modern:’ represented with Yahoo Web documents in Figure 2, positive examples are constructed from the examples included in 3 subtrees (‘Groups’, ‘Regional Information’, ‘Videos’) and 11 items containing hyperlinks to 11 actual documents.

When propagated to the higher levels, examples are assigned weight proportional to the size of node they appear in. We are using items on Yahoo documents that contain hyperlinks to actual Web documents as training examples instead of using actual Web documents. Our assumption here is that an item contains words representative for the document it points to. Words in each item represent description of the document it points to carefully chosen by the document author. In this way we reduce the time and space needed to collect and store training data. The actual Web documents are used as testing examples selected randomly from actual Web document (not Yahoo Web documents that are used in learning) accessible from the Yahoo hierarchy. The

final result of learning is a set of specialized classifiers. Classification of a new example includes consulting a specialized classifier for each of the categories. Each example is represented here as a feature-vector and each category classifier is represented with probability distribution over features. In the classification, category contributes only probability of features that are included in testing example. In case that all these features have zero probability, the category classifier returns prior probability over class values, that is negative class value. We can see that additionally to the prior probability of class value, only the features that are common to the example and category under consideration contribute to the classification result. The goal of classification using this set of specialized classifiers is to find categories that example potentially belongs to. We reduce the number of categories to be considered in the classification to less than a half by using inverted index. Namely, we assign to each feature a list of categories it appears in (has a probability  $> 0$ ). For each new example all the categories that share less than a required number of features with the example are pruned and only the remaining categories enter the process of classification.



Figure 2: Yahoo Web document representing category ‘Arts: Performing\_Arts: Dance: Modern’. The category is represented as a tree node that contains 11 actual documents and 3 subtrees. The last subtree ‘Videos’ is a pointer to category ‘Business and Economy: Companies: Arts and Craft: Performing Arts: Dance: Videos: Modern’.

Learning algorithm used in our experiments is naive Bayesian classifier on feature-vector as described in [6]. In this approach documents are represented with frequency vectors where an attribute is defined for each word position in the document hav-

ing word at that position as its value. The assumption about feature independence used in naive Bayesian classifier is here incorrect, especially when features representing several words are added. According to [2] this does not necessary mean that classifier will have poor performance because of that. Probability of class value for a given document is calculated as  $P(C|Doc) = \frac{P(C)\prod_j P(W_j|C)^{TF(W_j,Doc)}}{\sum_i P(C_i)\prod_i P(W_i|C_i)^{TF(W_i,Doc)}}$ , where  $P(W_j|C) = \frac{1+TF(W_j,C)}{\#words + \sum_i TF(W_i,C)}$ . Where  $TF(W_j,C)$  is the frequency of word  $W_j$  in documents having class value  $C$ ,  $\#words$  is the number of all words used in domain document representation and  $TF(W_j,Doc)$  is the frequency of word  $W_j$  in document  $Doc$ . In case of  $TF(W_j,Doc) = 0$ , meaning the word  $W_j$  does not occur in document  $Doc$ , the probability of word  $W_j$  does not influence the classification result. This property of used classifier enables pruning of unpromising categories.

## 4 Experimental results

Experimental evaluation of developed approach is performed using our recently developed machine learning system **Learning Machine** [4] that supports usage of different machine learning techniques on large data sets with especially designed modules for learning on text and collecting data from the Web. In our experiments we observe influence of the number of required common words used for pruning and the number of selected features (vector size in feature-vector document representation) to the categorization results over independent set of 100—300 testing examples selected randomly from the actual Web documents accessible from Yahoo hierarchy. We vary the number of required common features from 0 meaning no pruning to 20 meaning require that at least 20 features contained in the testing example have probability  $> 0$  in the category. We also vary the vector size from including 0.5 to 20 times the number of features that occur in positive examples. Notice that the number of features in positive examples is from just few to several tens that is in our domains much smaller than the number of all features. We get our categorization results from the set of independent classifiers, each potentially having different number of features. Thus we express the vector size in percentage of the number of features in positive examples, since the set of negative examples is the same for all classifiers. In this way, a classifier for a larger category is using more features than a classifier for some smaller category, while both classifying the same testing example. For each testing example we observe a list of categories each assigned a probability as a result of consulting a corresponding subproblem classifier. Sorting categories according to that probability gives ranking that we use to get the rank of the correct category. In case of several categories having the same probability, the ranking of the middle one is taken. In order to enable operational usage of the system on larger text domains we include pruning mechanisms. When classifying testing example only categories that share at least required number of features with the example are considered.

We report rank and probability assigned to the correct category, with rank 1 and probability 1 being the best result. To get summary results over testing examples we give mediana, since some of the testing examples are rather non-typical for their category making mean value inappropriate estimate of model quality. For instance, welcome page, page containing only one sentence asking for language preference, page containing error message or page giving redirection.

Results of naive Bayesian classifier for different vector sizes are given on domains representing three of the top fourteen Yahoo categories. ‘References’ having 129 nodes in the hierarchy and 923 (697+196+27+3+0) features, ‘Education’ having 349 nodes in the hierarchy and 3,215 (1,928+1,067+186+28+6) features and ‘Computers and Internet’ having 2652 nodes and 7,631 (5049+2276+261+38+7) features. We test our approach on three domains that are parts of the Yahoo hierarchy, since they are hierarchies themselves with the same problem characteristics as the whole Yahoo hierarchy but smaller.

Domain name	Vector size	Rank of correct category	Probability of correct category	Number of common features required
References	0.5	3	> 0.99	0-14
	1	3	> 0.99	0-10
	2	3	> 0.99	0-14
	<b>3</b>	<b>2</b>	> 0.99	<b>0-16</b>
	<b>4</b>	<b>2</b>	> 0.99	<b>0-14</b>
Education	0.5	3	> 0.99	0-5
	1	4	> 0.99	0-3
	2	3	> 0.99	0-3
	<b>3</b>	<b>3</b>	> 0.99	<b>0-16</b>
	<b>4</b>	<b>3</b>	> 0.99	<b>0-16</b>
	<b>6</b>	<b>3</b>	> 0.99	<b>0-14</b>
Computers and Internet	0.5	5	> 0.99	0-10
	1	7	> 0.99	0-2
	2	5	> 0.99	0-12
	3	5	> 0.99	0-12
	4	4	> 0.99	0-12
	<b>6</b>	<b>3</b>	> 0.99	<b>0-2, 8-12</b>
	<b>8</b>	<b>3</b>	> 0.99	<b>0-5, 8-12</b>
	10	3.5	> 0.99	0-2

Table 1: Result of experiments on three domains formed from Yahoo text hierarchy.

Figures 3, 4, 5 give mediana of rank and probability assigned to the correct category on three domains we used here. In all domains the best performance is achieved when only a small number of features is used enabling pruning of more than a half categories without performance degradation (see also Table 1). The best performance in rank and probability is achieved on for vector size ‘References’: 3-4, ‘Education’: 2-6, ‘Computers and Internet’: 6-8 times the number of features in positive class. Probability assigned to the correct category is for the best results over 0.99. More specifically, on ‘References’ is the mediana of the correct category rank 2 ie. the half of the testing examples are assigned rank 1 or 2 On ‘Education’ and ‘Computers and Internet’ the mediana of the correct category rank is 3 ie. the half of the testing examples are assigned rank 1, 2 or 3 for the correct category. On ‘References’ for the most vector sizes the lower quartile for rank and probability is 1, meaning that the fourth of the testing examples are assigned the highest rank to the correct category and the probability 1. On ‘Education’ and ‘Computers and Internet’ the lower quartile for rank is 1 or 2 and for probability 1.

Figures 6, 7, 8 show the probability of pruning the correct example category before even considering it in classification and the average number of considered categories. Both values change wiht the level of pruning and have almost the same value for all but the shortest vector size. The number of considered categories is reduced to more

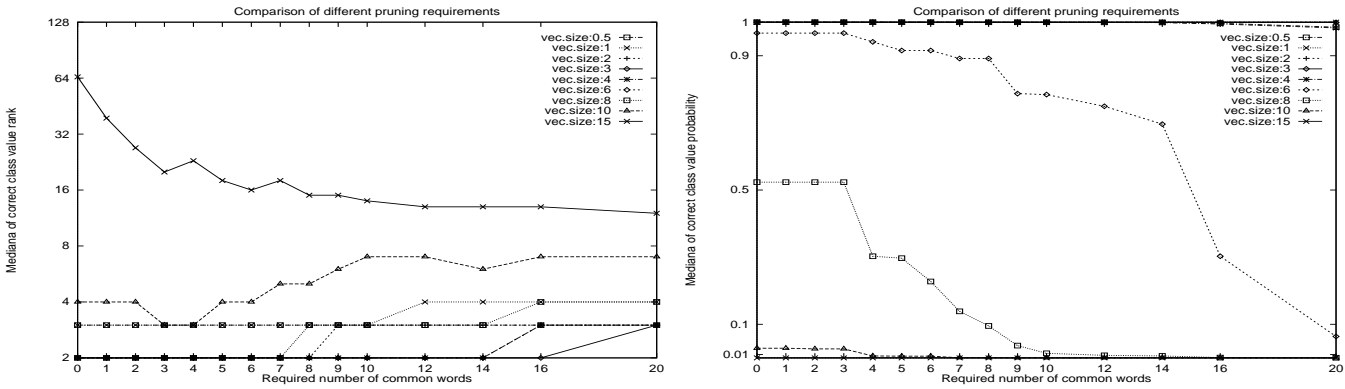


Figure 3: Comparison of different vector size when pruning unpromising categories on References (mediana of: correct category rank, probability of correct category).

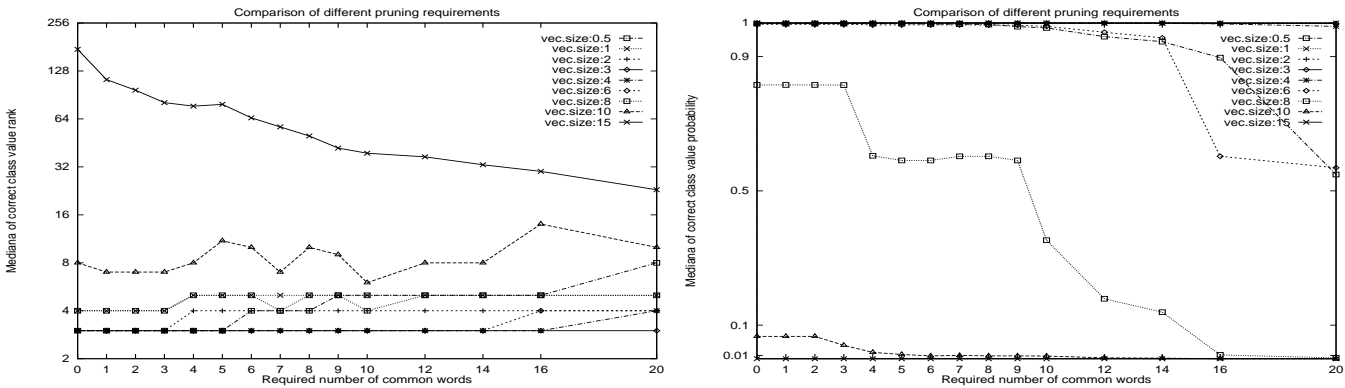


Figure 4: Comparison of different vector size when pruning unpromising categories on Education (mediana of: correct category rank, probability of correct category).

than a half while for 10% to 15% of testing examples the correct category is pruned but the mediana of rank and probability remains the same. More specifically, on ‘References’ for pruning level of 12 only about 40 out of 129 categories are considered, on ‘Education’ for pruning level of 14 only about 120 out of 349 categories are considered and on ‘Computers and Internet’ for pruning level of 12 only about 700 out of 2652 categories are considered. This pruning enables efficient text categorization on large text hierarchy.

## 5 Discussion

We have used several mechanisms to enable efficient text categorization. First, the new features representing word sequences are constructed after removal of common and infrequent words. Feature subset selection is then applied, where we report results for different subset size (vector size). Probability distribution over features is constructed for each category by propagating examples contained in the hierarchy subnode rooted at the node representing category. In this process features with low (less than 0.001) probability are pruned. The resulting probability distributions are used by naive Bayesian classifier. In the classification process the classifier for each category is applied. Each

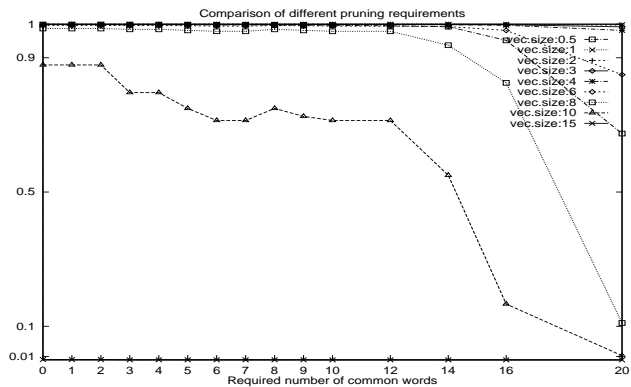
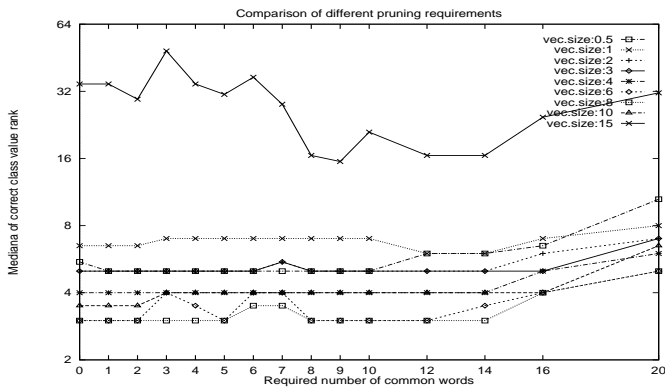


Figure 5: Comparison of different vector size when pruning unpromising categories on Computers and Internet (mediana of: correct category rank, probability of correct category).

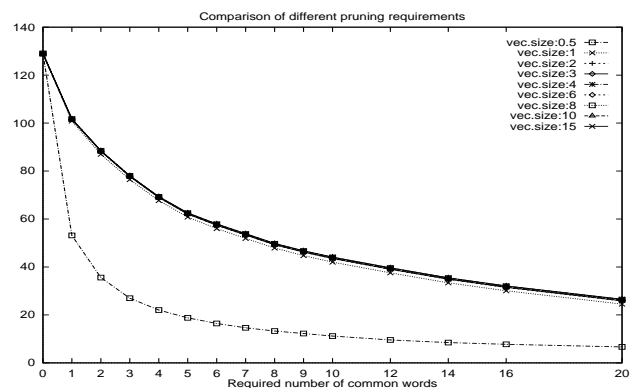
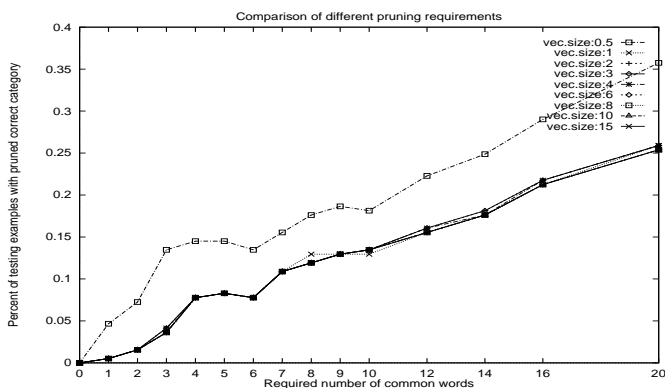


Figure 6: Comparison of different vector size when pruning unpromising categories on References (percent of testing examples with pruned correct category, average number of considered categories.)

classifier considers only probability of features that appear in the testing example. At that point we prune all the categories that share less than the required number of features with the testing example. Approach described in this paper enables text categorization on large text hierarchy represented with several thousands of features and several thousands of categories.

### Acknowledgements

This work was financially supported by the Slovenian Ministry for Science and Technology. Part of this work was performed during the authors stay at Carnegie Mellon University. Author is grateful to Tom Mitchell and his machine learning group at Carnegie Mellon University for generous support, suggestions and fruitful discussions.

### References

[1] Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I., 1996. Fast Discovery of Association Rules, In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth,

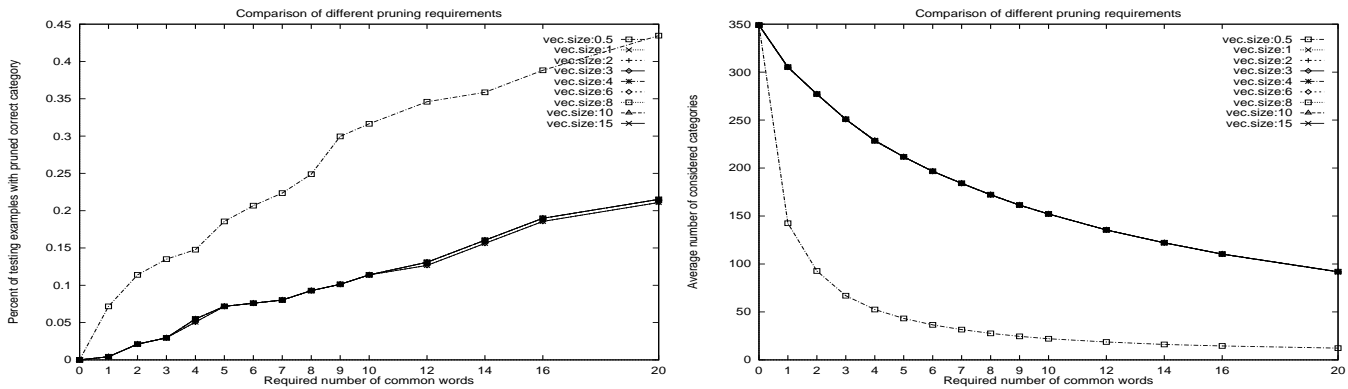


Figure 7: Comparison of different vector size when pruning unpromising categories on Education (percent of testing examples with pruned correct category, average number of considered categories.)

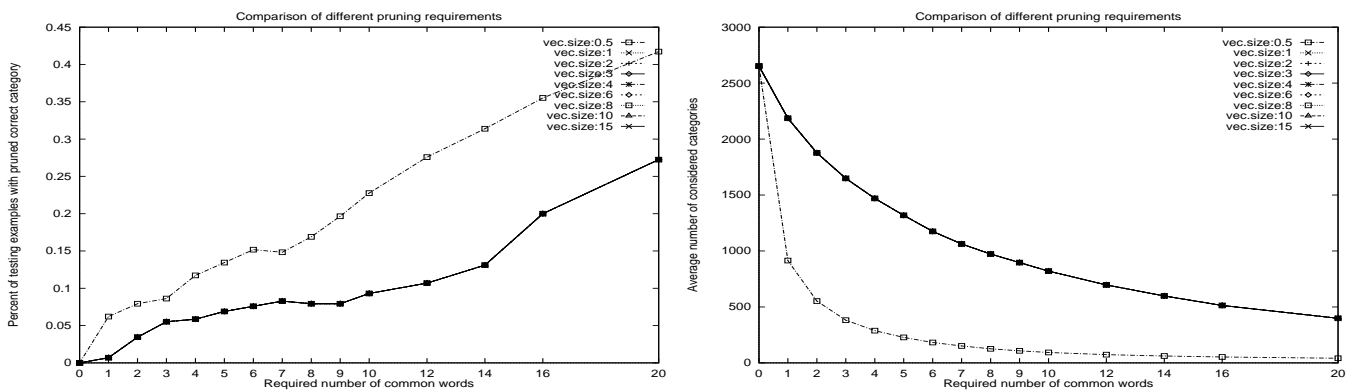


Figure 8: Comparison of different vector size when pruning unpromising categories on Computers and Internet (percent of testing examples with pruned correct category, average number of considered categories.)

P., Uthurusamy, R. (eds.), *Advances in Knowledge Discovery and Data Mining* AAAI Press/The MIT Press, pp. 307—328.

- [2] Domingos, P., Pazzani, M., 1997. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss, *Machine Learning 29*, Kluwer Academic Publishers, pp. 103—130.
- [3] Filo, D., Yang, J., 1997. Yahoo! Inc., 1997. <http://www.yahoo.com/docs/pr/>
- [4] Grobelnik, M., Mladenić, D., Learning Machine: design and impelmentation, *Technical Report IJS-DP-7824*, Department for Intelligent Systems, J.Stefan Institute, 1998.
- [5] Koller, D., Sahami, M., Hierarchically classifying documents using very few words, *Proc. of the 14th International Conference on Machine Learning ICML97*, pp. 170—178, 1997.
- [6] Mitchell, T.M., Machine Learning, The McGraw-Hill Companies, Inc., 1997.

- [7] Mladenić, D., Feature subset selection in text-learning, *Proc. of the 10th European Conference on Machine Learning ECML98*, 1998.
- [8] van Rijsbergen, C.J., Harper, D.J., Porter, M.F., The selection of good search terms, *Information Processing & Management*, 17, pp.77—91, 1981.
- [9] Shaw Jr, W.M., Term-relevance computations and perfect retrieval performance, *Information Processing & Management*, 31(4), pp.491—498, 1995.
- [10] Yang, Y., Pedersen, J.O., A Comparative Study on Feature Selection in Text Categorization, *Proc. of the 14th International Conference on Machine Learning ICML97*, pp. 412—420, 1997.