Mapping documents onto Web page ontology

Dunja Mladenić[1,2] and Marko Grobelnik[1]

[1] J.Stefan Institute, Ljubljana, Slovenia
{Dunja.Mladenic, Marko.Grobelnik}@ijs.si,
http://www-ai.ijs.si/{DunjaMladenic, MarkoGrobelnik}/
[2] Carnegie Mellon University, Pittsburgh, PA, USA
Dunja.Mladenic@cs.cmu.edu,
http://www.cs.cmu.edu/~dunja/

Abstract. The paper describes an approach to automatically mapping Web pages onto ontology using document classification based on the Yahoo! ontology of Web pages. Techniques developed for learning on text data are used here on the hierarchical classification structure (ontology of Web documents). The high number of features is reduced by taking into account the hierarchical structure and using feature subset selection developed for the Naive Bayesian classifier. We focus on data sets with many features that also have a highly unbalanced class distribution. Documents are represented as word-vectors that include word sequences of up to five consecutive words. Based on the hierarchical structure the problem is divided into subproblems, each representing one on the categories included in the Yahoo! hierarchy. The resulting model is a set of independent classifiers, each used to predict the probability that a new document is a member of the corresponding category represented as a node in the hierarchy. Our example problem is automatic document categorization where we want to identify documents relevant for the selected category. Usually, only about 1%-10% of examples belong to the selected category. Experimental evaluation on real-world data shows that the proposed approach gives good results. Our experimental comparison of eleven feature scoring measures show that considering data and algorithm characteristics significantly improves the performance.

1 Introduction

Web page ontology can be defined in different ways depending on the objective of the ontology. This Chapter addresses ontology of Web documents, where the documents themselves are organized in a content hierarchy, with more general nodes placed closer to the root of the hierarchy. Each node is labeled by a set of keywords describing the content of documents that are placed in the node. Each document is described by a one-sentence summary including a hyperlink that points to the actual Web document located somewhere on the Web. As manual construction and maintenance of such an ontology is time consuming, we have addressed a related subproblem of automatically assigning a set of the most relevant nodes from the existing ontology to a new text document.

For this task we have adopted text mining methods in particular using machine learning classification algorithm Naive Bayes on text data, where the data is obtained from the publicly available Web page ontology Yahoo [1]. The classification algorithm is applied on word-vector representation of Web documents, where each word from the document (also referred to as an example) corresponds to a vector component (usually referred to as a feature). In general, features used to describe examples are not necessarily all relevant and beneficial for the task in hands. Additionally, a high number of features may slow down the process while giving similar results as obtained with a much smaller feature subset. Feature subset selection is commonly used when classifying text data, since most text data is characterized by several tens of thousands of features. Moreover, our experiments show that using the proposed feature subset selection improves the performance of our approach enabling better mapping of Web pages on the existing ontology.

Architecture of the system is described in Section 2. Section 3 describes the process of constructing word sequences that are used for describing content of text documents. Section 4 gives description of the process used to select subsets of words using several known and some new scoring measures. Data characteristics are given in Section 5. Experimental comparison of the tested measures on real-world data is given in Section 6. The results are discussed in Section 7.

2 Architecture of the system

Input to our system is an ontology of Web documents, where the ontology contains only hyperlinks to the Web documents while the documents themselves are located all over the Web. The ontology is organized as a content hierarchy, with more general nodes placed closer to the root of the hierarchy (see Section 5 for data description). The ontology itself is represented with a set of Web documents located at the ontology site, here referred to as Yahoo Web documents. For each ontology node there is a Yahoo Web document containing:

- the node name - a set of keywords describing the content of documents that are placed in the node,
- a set of hyperlinks to the other ontology nodes,
- a set of hyperlinks to the actual Web documents with a short content description for each of them.

Output of the system is a model represented as a set of classifiers. The model is used for inserting a new Web document into the ontology, so that for a new Web document it returns a ranked list of the best fitting ontology nodes. If the top node in the ranked list has very low score (low probability that the new documents belongs to it), this indicates the need for extending the ontology. The whole system consists of four phases, as shown in Figure 1:

feature construction used to transform the text document given in html format to word-vectors as described in Section 3,

subproblem definition that takes the documents organized into ontology and forms a set of weighted training examples for each ontology node

feature selection used to reduce dimensionality of the training examples, this is specific to each node

classifier construction that for each node takes the set of associated training examples and constructs a classifier

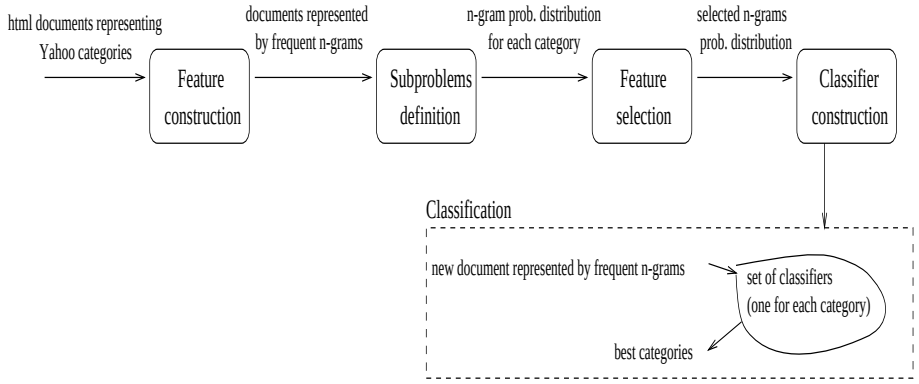

Fig. 1. Arichitecture of the system for automatically mapping documents onto an existing ontology.

The documents organized in the existing ontology are used as training examples for the classifier construction that models the mapping of documents into different parts of the ontology. The **feature construction phase** takes html documents, generates new features (word sequences) and represents each html document as a word-vector of that features. In the **subproblem definition phase** a binary classification problem is defined for each node in the ontology (hierarchy). Each problem is handled separately, applying **feature selection** and **classifier construction**. The final model consists of a set of classifiers, one for each ontology node. The model is used for classification (mapping) of a new document onto a set of nodes, so that we get a subset of the ontology nodes ranked according to the probability that the new document should be mapped onto each of them.

The classifier construction is performed using the Naive Bayesian classifier base on the multinomial event model as suggested in [2] Notice that the product goes over all word sequences that occur in the representation of document Doc.

$$P(C|Doc) = \frac{P(C)\Pi_{W_j \in Doc} P(W_j|C)^{TF(W_j,Doc)}}{\sum_i P(C_i)\Pi_{W_l \in Doc} P(W_l|C_i)^{TF(W_l,Doc)}}$$

Where $P(W_j|C)$ is the conditional probability here estimated using Laplace probability estimate, $TF(W_j, Doc)$ is the frequency of word W_j in document Doc. Notice that using word sequences instead of only single words means even stronger violation of the feature independence assumption used in the Naive Bayesian classifier.

3 Using word sequences

In our approach, each document is represented as a word-vector, where one vector component referred to as a feature, reflects the number of occurrences of a word or a word sequence in the document. Features are generated using an algorithm [3] similar to the Apriori algorithm [4] used for association rule generation.

We use word sequences, also known as n-grams, to form the new features. We generate features that represent up to five words (1-grams, 2-grams, ... 5-grams) occurring in a document as a sequence (eg. 'machine learning', 'world wide web'). We interleaved this feature generation with the feature selection using "stop-list" with common English words and removal of infrequent features. First we remove words contained in the "stop-list" (eg. a, the, for, each, at, will, be) enabling some features that represent a word sequence to actually capture longer sequences. For instance, 'Word for Windows' that is represented as a 2-gram 'Word Windows' or 'winners will be posted at the end of each two-week period' that is represented as a 5-gram 'winners posted end two-week period'. We additionally reduce the high number of features by pruning infrequent features. Infrequent features are here defined as features (word sequences) that occur less than 4 times. By using word sequences we can capture some characteristic word combinations and also increase the number of features. For example, in the whole Yahoo hierarchy (without a top category *'Regional'*) that we use for automatic document categorization the number of features has increased from 317,522 features representing single words (before removing infrequent features) to 1,410,303 features (before removing infrequent features) representing word sequences containing 1 or 2 words.

The process of feature generation is performed in passes over documents, where *i-grams* are generated in the i-th pass 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 [4]. In Figure 2 we give Algorithm for the generation of new features. In the first pass all single words not contained in the "stop-list" and having sufficient frequency (here we check for term frequency > 3) are taken `LargeNGramSet`. Word sequences of size 2 up to `MaxNGramSize` (here we take `MaxNGramSize` $= 5$) are generated using several pruning criteria. In each pass, all documents are checked one by one using a window `NGramQueue` to get a sequence of words. Before the next word is added to the window (a window is moved one word to the right) the word is checked for being a proper word (not a number or a special symbol), for not being in the "stop-list" `StopWordSet` and for being included in the current set of word sequences `LargeNGramSet`. The window is reset when the next word is not a proper word or is not included in the set of word sequences generated so far.

Variables used in Algorithm in Figure 2:
Input variables:

MinNGramOcc - minimal occurrences of N-Gram to become large N-Gram
MaxNGramSize - maximal size of N-Grams

StopWordSet - set of stop words (language dependent)
DocVec - vector of documents (DocVec[1], ..., DocVec[|DocVec|])
SymVec - vector of document lexical symbols (SymVec[1], ..., SymVec[|SymVec|])

Temporary variables:
Sym - lexical symbol in document (possible value types: word, number, symbol)
CandNGramMap - mapping from candidate N-Gram to its occurrence counter
NGramQueue - queue (window) of last NGramSize words (excluding StopWordSet)

Output variables:
LargeNGramSet - set of large N-Grams (occurring $\geq$ MinNGramOcc times)

Algorithm (C like pseudo-code):

```
1.  LargeNGramSet = all single words in DocVec, not in StopWordSet and
2.  occurring ≥ MinNGramOcc times;
3.  for NGramSize = 2 to MaxNGramSize do {
4.    CandNGramMap=[];
5.    for SymVec = DocVec[1] to DocVec[|DocVec|] do {
6.      NGramQueue=[];
7.      for Sym = SymVec[1] to SymVec[|SymVec|] do {
8.        if (TypeOf(Sym)==word){
9.          if (Sym not in StopWordSet){
10.           if Sym in LargeNGramSet then {
11.             if (|NGramQueue|+1==NGramSize){
12.               if (Concatenated(NGramQueue) in LargeNGramSet){
13.                 NGramQueue.Push(Sym);
14.                 CandNGramMap[Concatenated(NGramQueue)]++;
15.                 NGramQueue.Pop();
16.                     } else {NGramQueue.Push(Sym); NGramQueue.Pop();}
17.             } else {NGramQueue.Push(Sym);}
18.           } else {NGramQueue=[];}
19.         } /* if (Sym not in StopWordSet) */
20.       } else {NGramQueue=[];}
21.     }; /* for Sym */
22.   }; /* for SymVec */
23.   LargeNGramSet += {NGram:CandNGramMap[NGram] ≥ MinNGramOcc};
24. };
25. return LargeNGramSet;
```

Fig. 2. Algorithm for the generation of features representing word sequences (n-grams). Please notice that `NGramQueue` is a queue and not a stack. The generated features are used in enriched bag-of-words document representation.

For illustration, we show in Figure 3 the accumulated number of features during the described process of feature generation on Yahoo documents that compose the whole Yahoo hierarchy. The left hand-side graph shows for each pass of the *i-gram* generation the influence of the number of included Yahoo documents to the number of new features. For example, in the first pass (the lowest curve in the left hand-side of Figure 3) about 100,000 features are generated after including 20,000 documents, about 200,000 features after including 30,000 documents and about 320,000 features after including all 49,600 documents. After reducing the number of features by pruning infrequent features we get about 70,000 features representing single words (1-grams) to start with in the second pass over documents. In the second pass (the steepest curve in the left hand-side of Figure 3) all pairs of words (2-grams) that appear in the documents and consist of words included in the set of 1-grams are added. As can be seen from Figure 3, the second pass adds many infrequent features resulting with more than 1,400,000 features that after pruning infrequent features reduce to about 200,000 features. We stop with 5-grams since, in the fifth pass only about 10,000 features are added (the almost flat curve in the left hand-side of Figure 3) and about 6,000 of them are deleted as infrequent features. The right hand-side graph in Figure 3 shows the same process of feature generation over time. Influence of pruning infrequent features after each pass is even more evident here. It can be also clearly seen that the highest number of features is added in the second pass where 2-grams are generated, while in the fifth pass the curve showing the increasing number of features is almost flat (the right most part of the right hand-side graph in Figure 3).

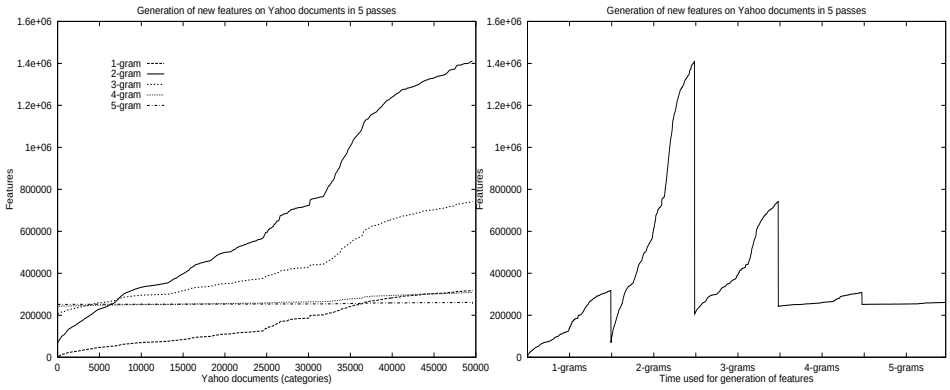

Fig. 3. Influence of included Yahoo documents to the accumulated number of features (left hand-side graph) and the process of feature generation (right hand-side graph). Notice that the two graphs represent the same results in two different ways. At the end of each pass over documents all the features that occur less than 4 times are deleted.

4 Selecting subset of words

Selecting subset of words to be used to describe content of documents is based on the idea of selecting subset of features commonly used in machine learning. When dealing with text data, we can easily have several tens of thousands of different words (features). Thus the whole process of feature subset selection is simplified by the assumption of feature independence. In this way the solution quality is traded for the time needed to find a solution, justified by the large number of features usually present in text data. Basically, a scoring is applied to each feature and the features are sorted according to the assigned score. Then, a predefined number of the best features is taken to form the feature subset.

Scoring of individual features can be performed using some of the measures used in machine learning for instance, $Information\ gain$ used in decision tree induction [5]. In our comparison eleven feature scoring measures were included. Information gain was included as the well known measure successfully used in some text-learning experiments. *Expected cross entropy* used in text-classification experiments [6] is similar to Information gain. The difference is that instead of calculating average over all possible feature values, only the value denoting that word occurred in a document is considered. Our experiments show that this means an important difference in the resulting performance. $Mutual\ information$ used in text-classification experiments [7] is similar to $Cross\ entropy\ for\ text$ with the latter additionally taking into account word probability. These two measures and a very simple frequency measure proposed in [7] were reported to work well on text data. This third measure is calculated either as the number of documents that contain word W referred to as document frequency $DF(W)$ or as the number of occurrences of word W referred to as $Term\ frequency\ TF(W)$. This measure requires the stop words removal. We use the second definition: $Freq(W) = TF(W)$. *Odds ratio* is commonly used in information retrieval, where the problem is to rank out documents according to their relevance for the positive class value using occurrence of different words as features [8].

$$OddsRatio(W) = \log \frac{P(W|pos)(1 - P(W|neg))}{(1 - P(W|pos))P(W|neg)}$$

Where $P(W|pos)$ is the conditional probability of word W occurring in documents belonging to the target content category and $P(W|neg)$ is the conditional probability of word W occurring in the other documents.

Our experiments show that *Odds ratio* is especially suitable to be used in a combination with the Naive Bayesian classifier for our kind of problems. We propose four variants of *Odds ratio*, to test if the results are sensitive to some modifications in the formula. As a baseline method we used random scoring method defined to score each word by a random number.

5 Data description

Data sets used in our experiments were generated from the Yahoo! hierarchy of Web pages as described in [9] and present five out of the fourteen sub-hierarchies that can be formed from the top level ontology nodes: *'Arts and Humanities'*, *'Entertainment'*, *'Computers and Internet'*, *'Education'*, *'References'*.

5.1 Web page ontology

The categories in Yahoo! ontology of Web pages are human constructed and designed for human Web browsing. Documents that are already classified and used to build the hierarchy are Web documents, making the hierarchy biased toward human knowledge areas that are represented in Web documents. For example, one of the most general categories named *'Bussines and Economy'* is over-represented including more than a third of all documents, while the other category that appears at the same level of the hierarchy named *'Social Science'* includes less than 1% of all documents. Our approach to mapping Web pages onto ontology is not limited to the Yahoo! hierarchy and can be used on some of the other text hierarchies that are not Web-oriented.

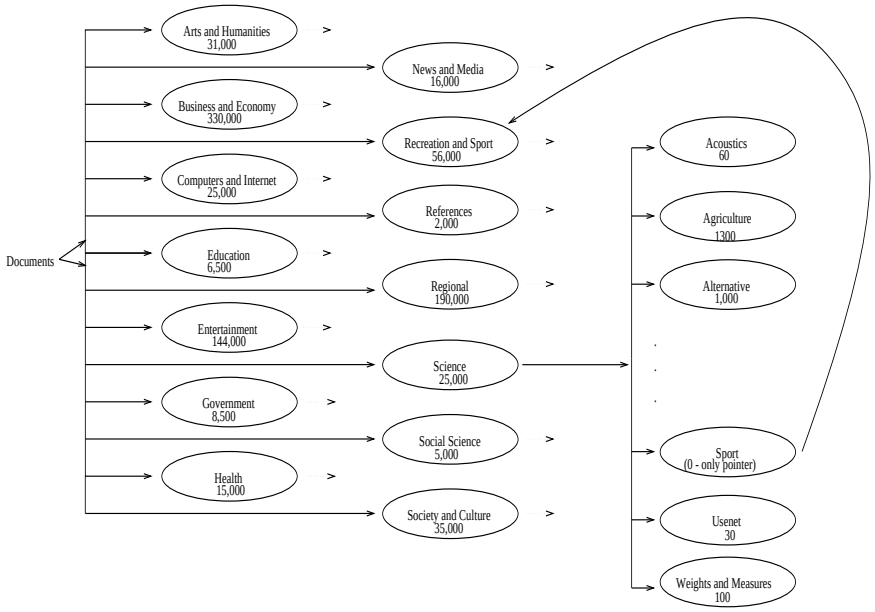

Fig. 4. Top level of the Yahoo categorization with the first level of subcategories in *'Science'* category. Approximate number of documents in each category is given under the category name (eg. 25,000 documents are classified under *'Science'*). The data is taken from the Yahoo site in UK & Ireland, November 1997.

The Yahoo! hierarchy when we obtained it (excluding subcategory *'Regional'*) was built on approximately a million Web documents located on Internet all around the world. We refer here to these documents as *actual Web documents*. Hyperlinks to those documents are organized in about 50,000 *Yahoo Web documents*. Each Yahoo document represents one of the included categories with the more general categories closer to the root of the hierarchy. The category is denoted by keywords, describing category content, that appear on the path from the root of the hierarchy to the node representing the category. For instance, *'Sport'* a subcategory of *'Science'* in Figure 4 is named *'Science: Sport'* and in our approach assigned two keywords: *Science, Sport*. In other words, a more specific category is named by adding a keyword to the name of the more general category directly connected to it (one level higher in the hierarchy).

Yahoo documents are connected with hyperlinks forming a hierarchical structure that can be represented as a directed acyclic graph (or a tree with additional connections between some of its nodes). For the sake of simplicity, we will talk about the tree and remind on additional connections only when needed. Each Web document classified in the Yahoo hierarchy appears only once, but there can be several connections in the hierarchy (hyperlinks between Yahoo documents) leading to it. For example, the Yahoo document representing *'Sport'*, a subcategory of *'Science'* includes a hyperlink to the top category *'Recreation and Sport'* (see Figure 4). In this way, the user still has the impression of a tree structure that nicely matches our intuition about the hierarchical organization of categories. Some nodes at the bottom of the hierarchy contain mostly hyperlinks to the actual Web documents, while the other nodes contain mostly or even only (eg. *'Science: Sport'* in Figure 4) hyperlinks to other Yahoo Web documents (nodes in the hierarchy). There are currently fourteen top level Yahoo categories each named by only one keyword. Figure 4 shows them with the approximate number of actual Web documents each category is based on. Each of the top categories is further represented with a hierarchical structure of more specific categories. As an example we show in Figure 4 the part of the first-level subcategories in *'Science'* top category ranging from *'Acoustics'* (named *'Science: Acoustics'* and assigned two keywords *Science, Acoustics*) to *'Weights and Measures'* (named *'Science: Weights and Measures'* and assigned two keywords *Science, Weights and Measures*).

5.2 Data sets used in our experiments

We have defined the problem as predicting document category based on the documents that are already categorized into a Web page ontology. Each data set is given as a classification hierarchy of text documents, with the more general categories closer to the root of the hierarchy. Each category is denoted by keywords, describing category content. More specific category is named by adding a keyword to the name of the more general category that is one level higher in the hierarchy.

The data was obtained from the publicly accessible Yahoo Web site. Table 1 gives data set characteristics including information about the number of nodes

in the hierarchy, the number of features showing also how many features represent different length word sequences, the number of examples (documents), the average distance between nodes measured as the average number of connections between any two nodes in the hierarchy and the average number of features in positive documents. The average number of features in positive documents is calculated as the average over the defined subproblems. Features are generated for word sequences (n-gram) of length up to fice consequtice words as described in Section 3.

Yahoo dataset	# categories (nodes)	# features (1-grams+...+5-grams)	# examples (Web docs)	Avg. node distance	Avg. pos. features
'Entertainment'	8,081	30,998 (15,144+11,211+ 2,970+1,059+505)	79,011	7.56	60
'Arts and Humanities'	3,085	11,473 (7,380+3,538+ 463+75+17)	27,765	6.59	65
'Computers and Internet'	2,652	7,631 (5,049+2,276+ 261+38+7)	23,105	6.77	55
'Education'	349	3,198 (1,919+1,061+ 184+28+6)	5,406	4.54	100
'References'	129	928 (701+196+ 28+3+0)	1,995	4.29	45

Table 1. Data set characteristics for five data sets formed from the five top Yahoo categories. The problem is document categorization based on the hierarchy of categories and the corresponding documents. For each data set we give (from left to right) its name, the number of included content categories, the number of features (word sequences and their length), the number of examples (actual Web document the category is based on), the average distance between two nodes in the hierarchy, the average number of features in positive examples (average number of word sequences in all the documents belonging to one category).

More specifically, the following data sets are used in our experiments. *'Entertainment'* having 8,081 nodes with an average distance 7.56 between them, 30,998 features consisting of 15,144 1-grams, 11,211 2-grams, 2,970 3-grams, 1,059 4-grams, 505 5-grams and 79,011 actual Web documents. *'Arts and Humanities'* having 3,085 nodes with an average distance 6.59 between them, 11,473 (7,380+3,538+463+75+17) features and 27,765 actual Web documents. *'Computers and Internet'* having 2,652 nodes with an average distance 6.77 between them, 7,631 (5,049+2,276+261+38+7) features and 23,105 actual Web documents. *'Education'* having 349 nodes with an average distance 4.54 between

them, 3,198 (1,919+1,061+184+28+6) features and 5,406 actual Web documents. *'References'* having 129 nodes with an average distance 4.29 between them, 928 (701+196+28+3+0) features and 1,995 actual Web documents.

From the data characteristics given in Table 1 it can be seen that the number of unique words (1-grams) varies from 701 for *'References'* to 15,144 for *'Entertainment'*. The set of negative examples is the same for all subproblems in one data set. The set of positive examples changes for each of the subproblems. For most of the subproblems the probability of positive class value is < 0.1, meaning that we have a problem with unbalanced class distribution.

In order to model a hierarchy of content categories and handle a large number of categories we generate a binary classification model for each of the hierarchy nodes (leaf or non-leaf) by collecting word probabilities from all the documents in and below the node (see [10, 11] for more information). These documents are taken as positive examples while all the other documents in the hierarchy are used as negative. This means that the most classifiers are induced from highly unbalanced class distribution with only about 1%-10% of positive examples. Since each classifier competes with the other classifiers, it is important that a classifier identifies documents belonging to its category. Thus we say that our problem has asymmetric misclassification costs given only implicitly in the problem. By asymmetric misclassification costs we mean that one of the class values (positive) is the target class value for which we want to get predictions and we prefer false positive over false negative.

6 Experimental results

In our experiments we compare different feature scoring measures and observe the influence of the number of selected features to the system performance. Since we have a set of classifiers, the number of selected features is determined relatively to the classifier category size expressed by the number of features in positive examples. We refer to this relative number of features as *Vector size*. 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. Reported results are averaged over 5 repetitions using hold-out testing on independent set of 500 (300 for the two smaller data sets) randomly selected testing examples. In order to enable operational usage of the system on larger data sets we include the pruning mechanisms described in [12]. Result is speed up of the classification process, since for each document classification, not all but only promising categories are considered (85%-95% of all categories are pruned).

Dom. name	Scoring measure	Average on keyword assignment			
		F1-measure	F2-measure	Precision	Recall
Ent.	**Odds ratio**	**0.48** ± 0.006	**0.59** ± 0.007	**0.44**± 0.006	**0.80** ± 0.006
	Term frequency	0.39 ± 0.003	0.49 ± 0.12	0.41± 0.006	0.71 ± 0.010
	Cross entropy Txt	0.29 ± 0.007	0.39 ± 0.007	0.35± 0.003	0.69 ± 0.007
	Mutual info. Txt	0.25 ± 0.005	0.27 ± 0.006	0.57± 0.007	0.38 ± 0.007
	Information gain	0.27 ± 0.008	0.22 ± 0.004	0.86± 0.005	0.21 ± 0.006
	Random	0.002± 0.001	0.002± 0.001	0.99± 0.006	0.001± 0.007
Arts.	**Odds ratio**	0.46 ± 0.003	**0.59** ± 0.005	**0.40**± 0.002	**0.83** ± 0.006
	Term frequency	**0.47** ± 0.007	**0.58** ± 0.008	**0.48**± 0.003	**0.77** ± 0.009
	Cross entropy Txt	0.32 ± 0.003	0.44 ± 0.004	0.33± 0.004	0.75 ± 0.008
	Mutual info. Txt	0.31 ± 0.005	0.35 ± 0.004	0.56± 0.007	0.46 ± 0.006
	Information gain	0.25 ± 0.006	0.21 ± 0.005	0.94± 0.002	0.20 ± 0.004
	Random	0.0051± 0.001	0.001± 0.001	0.99± 0.001	0.001± 0.001
Comp.	**Odds ratio**	0.46 ± 0.006	**0.60** ± 0.006	**0.40**± 0.007	**0.84** ± 0.005
	Term frequency	**0.48** ± 0.008	**0.58** ± 0.007	**0.50**± 0.004	**0.74** ± 0.005
	Cross entropy Txt	0.37 ± 0.007	0.49 ± 0.008	0.35± 0.004	0.75 ± 0.007
	Mutual info. Txt	0.36 ± 0.003	0.38 ± 0.003	0.62± 0.005	0.45 ± 0.005
	Information gain	0.21 ± 0.005	0.17 ± 0.005	0.94± 0.005	0.15 ± 0.005
	Random	0.01 ± 0.001	0.01 ± 0.001	0.99 ± 0.001	0.004± 0.001
Edu.	**Term frequency**	**0.48** ± 0.007	**0.55** ± 0.007	**0.57**± 0.010	**0.65** ± 0.010
	Odds ratio	0.33 ± 0.008	**0.48** ± 0.008	**0.36**± 0.010	**0.81** ± 0.005
	Mutual info. Txt	0.40 ± 0.004	0.46 ± 0.007	0.48± 0.010	0.59 ± 0.010
	Cross entropy Txt	0.32 ± 0.009	0.46 ± 0.007	0.28± 0.010	0.82 ± 0.006
	Information gain	0.13 ± 0.007	0.11 ± 0.006	0.98± 0.002	0.11 ± 0.006
	Random	0.01 ± 0.002	0.01 ± 0.002	0.99± 0.001	0.003± 0.002
Ref.	**Odds ratio**	**0.53** ± 0.006	**0.64** ± 0.006	**0.51**± 0.007	**0.81** ± 0.008
	Cross entropy Txt	0.52 ± 0.008	**0.60** ± 0.010	**0.62**± 0.003	**0.71** ± 0.010
	Mutual info. Txt	0.52 ± 0.010	0.55 ± 0.010	0.73± 0.010	0.60 ± 0.020
	Term frequency	0.50 ± 0.010	0.53 ± 0.010	0.78± 0.005	0.57 ± 0.010
	Information gain	0.25 ± 0.007	0.22 ± 0.007	0.99± 0.002	0.21 ± 0.006
	Random	0.07 ± 0.006	0.06 ± 0.005	0.99± 0.001	0.05 ± 0.005

Table 2. Comparison of feature scoring measures for the problem of keyword prediction on five data sets formed from the Yahoo hierarchy. For each data set, the compared feature scoring measures are sorted according to their performance in F2-measure. The values of F1-measure, Precision and Recall are given for better understanding. We give averages with standard errors calculated over 5 runs.

Dom. name	Scoring measure	Average on category prediction			
		F1-measure	F2-measure	Precision	Recall
Ent.	**Odds ratio**	**0.29**± 0.002	**0.30** ± 0.003	**0.41** ± 0.004	**0.34** ± 0.003
	Term frequency	0.24± 0.003	0.27 ± 0.003	0.38 ± 0.003	0.34 ± 0.003
	Mutual info. Txt	0.22± 0.004	0.23 ± 0.004	0.57 ± 0.006	0.29 ± 0.007
	Information gain	0.25± 0.002	0.20 ± 0.003	0.87 ± 0.002	0.17 ± 0.002
	Cross entropy Txt	0.15± 0.002	0.18 ± 0.002	0.29 ± 0.005	0.28 ± 0.005
	Random	0.001± 0.0001	0.001± 0.0002	0.99 ± 0.007	0.001±0.0002
Arts.	**Odds ratio**	**0.29**± 0.002	**0.32** ± 0.004	**0.36** ± 0.005	**0.38** ± 0.004
	Term frequency	0.28± 0.002	0.29 ± 0.003	0.43 ± 0.004	0.34 ± 0.003
	Mutual info. Txt	0.24± 0.005	0.25 ± 0.005	0.56 ± 0.006	0.31 ± 0.007
	Cross entropy Txt	0.18± 0.002	0.22 ± 0.003	0.27 ± 0.005	0.32 ± 0.006
	Information gain	0.21± 0.002	0.17 ± 0.002	0.93 ± 0.003	0.15 ± 0.002
	Random	0.0012± 0.0001	0.001± 0.0003	0.99 ± 0.006	0.001±0.0002
Comp.	**Odds ratio**	**0.30**± 0.002	**0.33** ± 0.002	**0.36** ± 0.009	**0.57** ± 0.005
	Term frequency	0.27± 0.003	0.26 ± 0.002	0.45 ± 0.003	0.27 ± 0.003
	Mutual info. Txt	0.25± 0.003	0.24 ± 0.004	0.60 ± 0.006	0.26 ± 0.006
	Cross entropy Txt	0.19± 0.005	0.21 ± 0.004	0.28 ± 0.004	0.27 ± 0.002
	Information gain	0.19± 0.007	0.14 ± 0.006	0.94 ± 0.004	0.12 ± 0.005
	Random	0.001± 0.0003	0.001± 0.0002	0.99 ± 0.001	0.001± 0.0002
Edu.	**Mutual info. Txt**	**0.40**± 0.006	**0.45** ± 0.009	**0.52** ± 0.010	**0.53** ± 0.010
	Odds ratio	0.32± 0.005	**0.43** ± 0.005	**0.36** ± 0.009	**0.57** ± 0.010
	Term frequency	**0.40**± 0.010	**0.42** ± 0.010	**0.57** ± 0.010	**0.45** ± 0.020
	Cross entropy Txt	0.2 ± 0.006	0.26 ± 0.004	0.23 ± 0.010	0.37 ± 0.005
	Information gain	0.09± 0.003	0.07 ± 0.002	0.97 ± 0.003	0.07 ± 0.002
	Random	0.01± 0.001	0 ± 0.001	0.99 ± 0.002	0.001± 0.001
Ref.	**Odds ratio**	**0.37**± 0.007	**0.42** ± 0.009	**0.46** ± 0.009	**0.51** ± 0.012
	Mutual info. Txt	0.34± 0.005	0.32 ± 0.005	0.69 ± 0.015	0.32 ± 0.006
	Term frequency	0.28± 0.007	0.26 ± 0.070	0.72 ± 0.007	0.26 ± 0.007
	Cross entropy Txt	0.23± 0.005	0.22 ± 0.005	0.50 ± 0.003	0.23 ± 0.005
	Information gain	0.20± 0.003	0.16 ± 0.002	0.99 ± 0.002	0.14 ± 0.002
	Random	0.05± 0.006	0.04 ± 0.005	0.99 ± 0.001	0.04 ± 0.004

Table 3. Comparison of feature scoring measures for the problem of category prediction on five data sets formed from the Yahoo hierarchy. For each data set, the compared feature scoring measures are sorted according to their performance in F2-measure. The values of F1-measure, Precision and Recall are given for better understanding. We give averages with standard errors calculated over 5 runs.

To evaluate the results we use Precision, Recall and F_2-measure as commonly used evaluation measures for text data [13]. $F-measure$ is a combination of Precision P and Recall R commonly used in information retrieval $F_\beta = \frac{(1+\beta^2)P \times R}{\beta^2 P + R}$. The relative importance of each is expressed with the value of parameter β. We report average Precision and Recall per document calculated for the fixed probability threshold (experimentally set to 0.95 [12]). Precision can be seen as the classification accuracy calculated only for positive examples, while Recall is the proportion of positive examples the system recognized as positive (values in [0..1]). If testing example is originally assigned to several categories, all these categories are taken as correct and compared to the set of predicted categories. We perform this comparison in two ways: (1) keyword prediction taking into account proximity to the correct category using keywords assigned to each category and (2) category prediction requesting prediction of the correct category.

Additional to Precision and Recall we report F_2-measure that is a combination of the two, commonly used when we care more about Recall than about Precision. Tables 2 and 3 give results of the comparison (1) and (2) respectively. We observe performance (F2-measure) for the best performing number of selected features that is in most cases vector size 1 (meaning select as many features as there are features that occur in positive examples). To get an idea about the actual number of the used features, vector size 1 means in average over categories: on *'Entertainment'* 58 out of 30,998 features (0.2%), on *'Arts and Humanities'* 65 out of 11,473 features (0.7%), on *'Computers and Internet'* 42 out of 7,631 features (0.62%), on *'Education'* 85 out of 3,198 features (2.7%), on *'References'* 49 out of 928 features (5.3%). Additionally to the value of F_2-measure, we give values of Precision, Recall and F_1-measure. F_1-measure is included to show how the same feature scoring measures would compare in case we would have a problem where Precision and Recall are equally important. As we can see from Tables 2 and 3, there would not be much difference, Odds ratio and Term frequency would remain the best.

On all data sets Odds ratio is among the best performing measures (see Tables 2 and 3) and the best performance is achieved when only a small number of features is used. For instance, in Table 2 on *'Computers and Internet'* Odds ratio achieves F_2-measure of 0.60, Precision of 0.40 and Recall of 0.84, meaning that 40 % of document predicted positive are positive and that 84% of all positive documents are identified. This is consistent with the results reported in text-learning on the problem of predicting clicked hyperlinks from the set of visited Web documents [14] where the feature selection based on Odds ratio achieved the best results. Similar observation regarding the number of features is reported on text categorization in [7] , where the reduction of up to 90% in the number of features resulted in either an improvement or no loss in the system performance. Observation of standard error on all five data sets confirms that the best performing measures are significantly better than the other tested measures.

Additionally, we report two non-standard but intuitive measures: rank and probability assigned to the correct category. For each testing example we observe a list of categories each assigned a probability as a result of consulting the

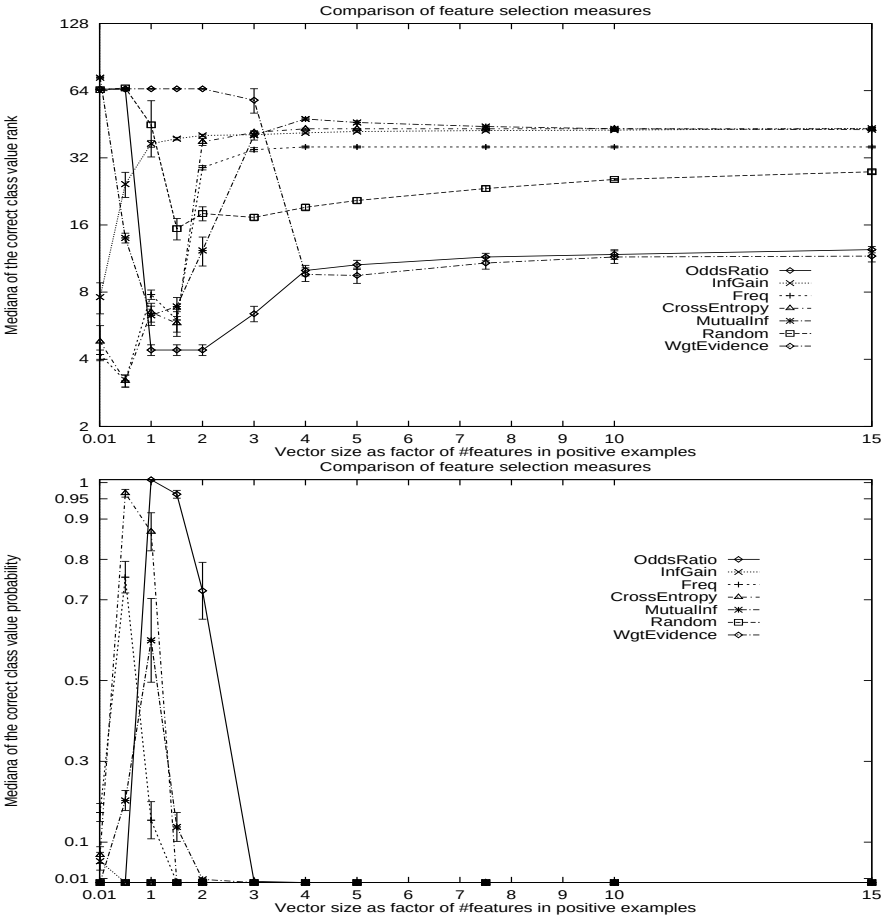

Fig. 5. Comparison of (upper) the correct category rank (the lower the better) and (lower) probability for different feature scoring measures on *'Reference'*.

corresponding subproblem classifier. Sorting categories according to that probability gives ranking that we use to get the rank of the correct category. If there are more correct categories, the one with the highest predicted probability is considered. To get summary results over the testing examples we give median rather than mean, since some of the testing examples are rather non-typical of their category, containing eg., a welcome page or only one sentence asking for language preference or an error message or a page giving redirection.

Rank and probability obtained in the same experiments show that Odds ratio is again the best or one of the two best performing measures. Cross entropy for text and Term frequency achieved similar results as Odds ratio on two out of the five data sets. Information gain performed similar as Random. Tables giving the detailed results of rank and probability can be found in [10]. Standard errors confirm that the results are significant. Observation of the average number of

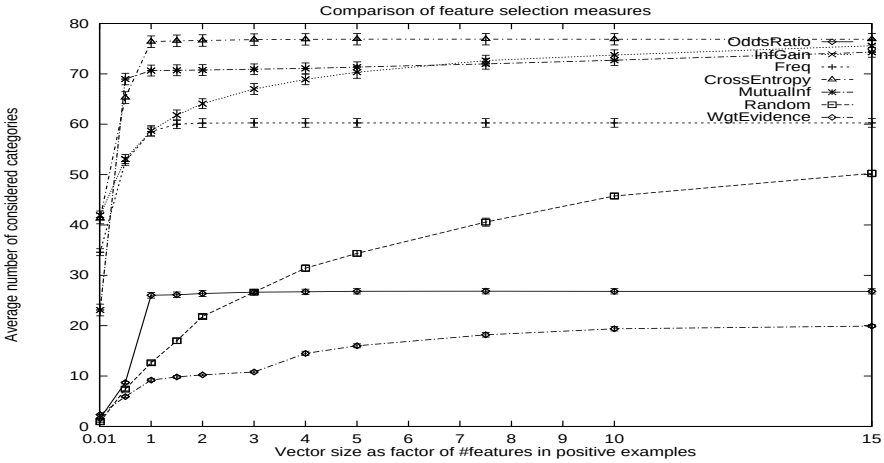

Fig. 6. Comparison of the number of considered categories (the lower the better) for different feature scoring measures on *'Reference'*.

considered categories during the classification shows that Odds ratio is considering about 3 times less categories than Cross entropy for text and about 2 times less categories than Term Frequency.

For illustration of the influence of the number of selected features we show graphs for *'Reference'*. Figure 5 gives median for the correct category rank and probability on *'Reference'*. It can be seen that the best performance in rank and probability is achieved by Odds ratio and Cross entropy for text using relatively small number of features (vector size 0.5-1.5). For instance, on *'References'* the mediana of the correct category rank is 3.8 the mediana of the correct category probability over 0.99, ie. the half of the testing examples are assigned rank up to 3.8 and probability > 0.99 for the correct category. For larger feature subsets the rank became insensitive to the additional features, while the probability is almost 0.

To show the efficiency of the used scoring measure we give in Figure 6 the influence of the relative number of selected features to the number of categories considered in classification. For Random scoring and Information gain this number grows with the growing number of selected features (vector size). For the other measures, the number of considered categories is mostly stable when using more features (eg., using ≥ 50 features for *'Reference'*). Odds ratio is the second best in the low number of considered categories and this is also one of the best performing measures. Cross entropy for text that is also among the best performing measures is the worst in the number of considered categories.

There is no significant difference in the performance between Odds ratio and any of its four variants we tested. This shows that the most important characteristics of Odds ratio are included in its variants. It also shows that we didn't get any significant improvement by including probability of word occurrence (Weighted Odds ratio) nor by including features characteristic for negative examples (Conditional Odds ratio).

7 Discussion

Mapping Web pages onto ontology is addressed here on a problem where text Web documents are organized into content hierarchy. There is some related work on automatic categorization of text documents. Koller and Sahami [6] proposed an approach to hierarchically classifying documents. They used the Reuters dataset that 'does not have a predetermined hierarchical classification structure', so they 'identified labels that tend to subsume other labels, and used those as the higher level topics'. In their hierarchy all documents are placed at the bottom of the hierarchy in tree leaves. Documents are represented as Boolean word-vectors with features representing words selected using greedy algorithm that eliminates features one by one using Cross entropy measure. They compared several learning algorithms and learn document category from the hierarchical structure, dividing classification task into a set of smaller problems corresponding to the splits in the classification hierarchy nodes. They give results on three domains each having a 3-level hierarchy that is based on up to 1,000 documents and the biggest having 12 nodes. McCallum et al. also developed an approach for learning from class hierarchies based on shrinkage [15] and reported results on the two bottom layers of the Yahoo hierarchy. However, there approach does not address the problem of documents being placed in any hierarchy node. Rather they used the same assumption as used in [6] that all the documents are places in the leaves.

The Yahoo data we are using here is much bigger, have predetermined hierarchical classification structure with document not necessarily placed only in the hierarchy leaves. This means that some documents sitting in the non-leaf nodes are too general to be classified into any of the existing leaf nodes. A classifier using these non-leaf documents should consider classification into any of the hierarchy nodes (not only leaf nodes). We use the Yahoo hierarchy for two learning problems: *assign categories* and *assign a set of keywords* to a document. For a new document, the learned model returns for each category (and the corresponding set of keywords) from in the text hierarchy the probability that the document is its member. One possibility is to use 'flattened' approach and generate one huge classifier with many class values, each value corresponding to one category. Additionally to the high number of class values, a large number of features is needed in order to cover all the variety of the large number of documents included in different categories. As pointed out in [6], an alternative idea is to split the whole problem into subproblems and use a local feature subset selection on each subproblem. We use the hierarchical structure to define subproblems each corresponding to the individual Yahoo category. For each of the subproblems, a classifier is constructed that predicts the probability for a document to be a member of the corresponding category and thus to be characterized by the corresponding set of keywords.

In our experiments with Naive Bayes the best performing feature scoring measures are Odds ratio and its variants, while the worst are Random and Information gain. The other perform comparable or worse than Odds ratio and better than Information gain. A closer look to the highly scored features by

Odds ratio and by Information gain explains the huge difference in their performance and poor performance of Information gain. It also indicates how important is to consider data set and algorithm characteristics. In related work, Brank et al. [16] report that on Reuters data set, feature selection using Odds ratio indeed improves the classification results of Naive Bayes but it does not help SVM-classifier, which performs better than Naive Bayes. Testing this on our data set is an interesting problem for future work.

Yang and Pedersen [7] give experimental comparison of five measures for feature selection in text categorization on word-vectors representing documents. Their experiments confirm our observation that a rather small feature subset should be used since it gives either better or as good results as large feature subset or all the features. We also agree in the observation that a simple frequency of a feature (used after stop words removal and calculated either using term or document frequency) achieves very good results. Our results disagree in the performance of Information gain. Information gain was one of the best performing measures on problems addressed by Yang and Pedersen [7], while we found it performing poorly (similar to random) on our data. The reason for that we find in the difference in our data and classification algorithms. Their data set is defined to include one class value for each category, while we split the problem into subproblems each corresponding to one category and having binary-valued class. The result of learning is in our case a set of specialized classifiers instead of one huge classifier including the union of features characteristic for different categories. The other important issue is the used classification algorithm. Yang and Pedersen [7] used k-Nearest Neighbor and Linear Least Square Fit mapping. The specific of the Naive Bayesian classifier we are using is that it considers only features that occur in a classification document. This means that highly scored features should be features that will probably occur in new documents.

Poor performance of Information gain can be explained by its symetric treatment of class values. Here we have unbalanced class distribution and also highly unbalanced feature value distribution. When calculating feature score we observe two values for each feature (word sequence): occurs or does not occur in a document. The prior probability that a word sequence occurs in a document $P(W)$ is rather small. Most of the features selected by Information gain are features with the majority feature value ($P(\overline{W})$ is high). If $P(\overline{W}) >> P(W)$ then the high value of Information gain in most cases means that the second part $P(\overline{W}) \sum_i P(C_i|\overline{W}) \log \frac{P(C_i|\overline{W})}{P(C_i)}$ of the Information gain formula is high. In other words, knowing that W does not occur in a document brings useful information about the class value. Intuitively, when classifying a new document, a better classification results are expected if the classification is based on words that occur in a document. It is possible that absence of some words in a document is very informative and this is taken into account by the new feature scoring measure we named Conditional Odds ratio. The problem is that the classification based mostly on the absence of words is usually harder and requires larger feature subset than the classification based on word occurrences. Cross entropy for text makes distinction between the feature values and achieves good results

in our experiments. Odds ratio and its variants achieve better results than Cross entropy for text mainly due to favoring features characteristic for positive examples (high $P(W|\text{`}pos\text{'})$). The other tested measures make no distinction between the class values. Moreover, Information gain makes no distinction between the feature values (it is using feature absence). Since we have unbalanced class distribution with over 90% of examples having negative class value, most of the features are characteristic for negative examples. This means that most of the features highly scored by Information gain are either informative when they do not occur in a document or they are characteristic for negative class value (just the opposite of Odds ratio!).

8 Conclusions

In analysis of our experiments of document categorization using Naive Bayes, we have observed that the best performing feature scoring measures makes difference between the class values. The best results are achieved by Odds ratio that assumes that a problem has a binary-valued class and one of the class values is the target class value (asymmetric misclassification costs). The next group of measures achieving good results all favor common features (Cross entropy for text, Term frequency). Mutual information for text differs from Cross entropy for text only in not favoring frequent features and achieves worse results. Information gain differs from Cross entropy for text only in using feature absence as well as feature presence and achieves poor results.

Our conclusion is that in general the most important characteristics of a good feature scoring measure for text are: favoring common features and considering data and algorithm characteristics. For learning algorithms that make difference between feature presence and absence, such as the Naive Bayesian classifier used here, it is important that a scoring measure also makes this difference. For the data sets with binary-valued class where one class value is the target class value, the most important characteristics of a good feature scoring measure is to make difference between the class values and favor features characteristic for the target class value. In this case, favoring common features is not an important issue. Namely, Weighted odds ratio that favors common features is not performing better than Odds ratio.

Instead of using a filtering approach to feature selection that ignores the learning algorithm or using the wrapper approach that uses the learning algorithm as a 'black-box', we suggest that data and algorithm characteristics are studied in advance. We applied the Naive Bayesian classifier to the data sets that have an unbalanced class and feature value distribution and asymmetric misclassification costs, where the minority class value is the target class value. Experimental comparison of different feature scoring measures used in feature selection shows that Odds ratio achieves the best results. Our classifier uses the same conditional probability as used in Odds ratio for scoring the features. In this way, the selected features are features expected to have the greatest influ-

ence to the posterior probability of class values returned by the Naive Bayesian classifier.

More precisely, let us consider data and algorithm characteristics to see which features should be selected. In our case the majority class value is negative ($P(neg) > P(pos)$). If we want to identify the positive documents then for such documents the Naive Bayesian classifier should assign higer probability to the positive class than to the negative class value ($P(pos|Doc) > P(neg|Doc)$). Based on the formula of Naive Bayes (see Section 2), we can see that this can be achieved only if the inside product is higher for the positive class value than for the negative class value ($\Pi_{W_j \in Doc} P(W_j|pos)^{TF(W_j,Doc)} > \Pi_{W_j \in Doc} P(W_j|neg)^{TF(W_j,Doc)}$). At the same time, Odds ratio favors words that are more common in positive documents that in negative documents ($P(W_j|pos) > P(W_j|neg)$). Having many such words selected for learning means, that we have good chances to get the above mentioned product in the classifier higher for the positive than for the negative class value. Thus our suggestion is to use Odds ratio for feature selection when using the Naive Bayesian classifier for modeling the documents.

Moreover, our experiments suggest that we should select as many best features as there are features that occur in the positive examples. Closer look to the features sorted according to Odds ratio show that this approximately means simply select all the features that occur in positive examples without performing any feature scoring. In general, we can conclude that on such problems with unbalanced class distribution and asymmetric misclassification costs, features characteristic for the positive examples should be selected.

References

1. Filo D., Y.J.: Yahoo! inc. In: http://www.yahoo.com/docs/pr/. (1997)
2. McCallum, A., N.K.: A comparison of event models for naive bayes text classifiers. In: Proceedings of the AAAI-98 Workshop on Learning for Text Categorization. (1998)
3. Mladenić, D., G.M.: Word sequences as features in text-learning. In: Proceedings of the Seventh Electrotechnical and Computer Science Conference ERK'98, Slovenia: IEEE section. (1998)
4. Agrawal R., Mannila H., S.R.T.H.V.A.: Fast discovery of association rules. In: Advances in Knowledge Discovery and Data Mining, U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.). (1996)
5. Quinlan, J.: Constructing Decision Tree. Morgan Kaufman Publishers (1993)
6. Koller, D., S.M.: Hierarchically classifying documents using very few words. In: Proceedings of the 14th International Conference on Machine Learning ICML97. (1997)
7. Yang, Y., P.J.: A comparative study on feature selection in text categorization. In: Proceedings of the 14th International Conference on Machine Learning ICML97. (1997) 412–420
8. van Rijsbergen C.J., Harper D.J., P.M.: The selection of good search terms. Information Processing and Management **17** (1981) 77–91

9. Mladenić, D., G.M.: Feature selection on hierarchy of web documents. Journal of Decission support systems **35** (2003) 45–87
10. Mladenić, D.: Machine learning on non-homogeneous, distributed text data. In: PhD thesis, University of Ljubljana, Slovenia, http://www.cs.cmu.edu/∼TextLearning/pww/PhD.html. (1998)
11. Mladenić, D.: Turning yahoo into an automatic web-page classifier. In: Proceedings of the 13th European Conference on Aritficial Intelligence ECAI'98. (1998)
12. Grobelnik M., M.D.: Efficient text categorization. In: Proceedings of the ECML-98 Workshop on Text Mining. (1998)
13. Lewis, D.: Evaluating and optimizating autonomous text classification systems. In: Proceedings of the 18th Annual International ACM-SIGIR Conference on Recsearch and Development in Information Retrieval. (1995)
14. Mladenić, D.: Feature subset selection in text-learning. In: Proceedings of the 10th European Conference on Machine Learning ECML98. (1998)
15. McCallum A., Rosenfeld R., M.T.N.A.: Improving text classification by shrinkage in a hierarchy of classes. In: Proceedings of the 15th International Conference on Machine Learning (ICML-98), Morgan Kaufmann, San Francisco, CA (1998)
16. Brank J., Grobelnik M., M.F.N.M.D.: Interaction of feature selection methods and linear classification models. In: Proceedings of the ICML-2002 Workshop on Text Learning, Sydney: The University of New South Wales (2002)