Lappeenranta University of Technology, Department of Information Technology

ITELLIGENT INTERNET DATA SEARCHING

The topic of the master's thesis has been confirmed by the Department Council of the Department of Information Technology on 25 August 2000

Examiner: Prof. Jan Voracek
Supervisor: Prof. Jan Voracek

Lappeenranta 6 November 2000

Anna Merkoulova
Nuijatie 3 B 9
01650 VANTAA
+358 40 7009416

TIIVISTELMÄ

Tekijä	Anna Merkoulova
Työn nimi	**Älykäs Internettietohaku**
Osasto	Tietotekniikan osasto
Vuosi	2000
Paikka	Lappeenranta, Finland

Diplomityö. Lappeenrannan teknillinen korkeakoulu. 71 lehteä, 17 kuvaa, 15 taulukkoa. Tarkastajana apulaisprofessori Jan Voracek.

Hakusanat: feature extraction, machine learning, text classification, text categorization

Tässä artikkelissa tekijä käsittelee automaatista tekstin luokittelua. Tämä on ”järkevän” Internet-hakukoneen rakentamisen päätehtävä. Artikkelissa vertaillaan erilaisia tekstin ominaisuuksien erottamisen keinoja, pohditaan miten saadaan opettaa hakukoneen tekstin sisällön pohjalta. Tekijä myös selostaa oman tekstin ominaisuuksien monisanaisen erottamiskeinon ja esittää keinon lineaarisen diskriiminantianalyysin laaditun lajittelijan perusteella tehdyn testauksen tulokset. Myös testitulokset, jotka on saatu mallikappaleen erotteluun valvotun tavallisen backpropagation - algoritmin käyttävän ja valvottaman koulutuksen käyttävän lajittelijan perusteella, on esitetty ko. artikkelissa.

ABSTRACT

Written by	Anna Merkoulova
Title	**Intelligent Internet Data Searching**
Department	Department of Information Technology
Year	2000
Place	Lappeenranta, Finland

Diploma Thesis, Lappeenranta University of Technology, 71 pages, 17 figures, 15 tables. Examiner Associate Professor Jan Voracek.

Keywords: feature extraction, machine learning, text classification, text categorization

In this thesis author approaches the problem of automated text classification, which is one of basic tasks for building Intelligent Internet Search Agent. The work discusses various approaches to solving sub-problems of automated text classification, such as feature extraction and machine learning on text sources. Author also describes her own multiword approach to feature extraction and presents the results of testing this approach using linear discriminant analysis based classifier, and classifier combining unsupervised learning for etalon extraction with supervised learning using common backpropagation algorithm for multilevel perceptron.

LIST OF SYMBOLS AND ABBREVIATIONS

l	length, number of letters in word taken into account
LI	"length of interest" number of words in word sequence
step	"step" used for feature extraction
DF	Document Frequency
FTP	File Transfer Protocol
HTML	Hypertext Markup Language
k-NN	k-Nearest Neighbour
LDA	Linear Discriminant Analysis
LSI	Latent Semantic Indexing
NN	Neural Network
SL	Similarity Level
SOM	Self Organizing Map
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TFIDF	Term Frequency Inverse Document Frequency

1. INTRODUCTION

Internet is a huge and constantly growing amount of information on various themes. Many people broadly use Internet after invention of the World Wide Web (WWW). WWW is a technology providing easy to use access to the information resources of the computers, connected to the Internet around the world. Users can access various information on WWW servers by browsing the Web. Because Internet is a huge amount of non-homogeneous distributed information resources, systems that allow user to find interesting information quickly are very popular. Research and development of such intelligent agents, designed to help user in his Internet browsing, now became a many developers' issue. Many existing publications about Information Retrieval and Machine Learning are calling attention to the new construction methods of such Intelligent Agents and Intelligent User Interfaces. These methods are based on combination the Information Retrieval and Machine Learning methods and applying them to the large sources of text data, in order to help user in Internet searching.

Operating with the Internet searching resources (with the help of web search agents) user spending great amount of time inspecting whether links found by the agent according to his query, match the subject or not. Searching for the same query in the predefined category of documents may reduce time spent on these operations. It's well known that all major Web search agents operate the same way: a gathering program explores the hyperlinked documents in the Web, foraging for Web pages to index. These pages are stockpiled by storing them in some kind of database or repository. Finally, a retrieval program takes a user query and creates a list of links to Web documents matching the words, phrases, or concepts in the query. Intelligent web search agent that can also assign the category name to the indexed page and allow user to search in the predefined

category of documents will significantly reduce the scope of the search and simplify the task.

In general, the problem of classification may be determined as the following: for each document from the document collection its relations to the set of pre-defined categories should be determined. In other words, the problem is to determine the subject that fits the content of the document (“sport”, “business”, “politics” etc.)
In this thesis we look at the main approaches to the automatic text classification within the general machine learning paradigm. Chapter 2 describes three questions that we find important: what representation to use for documents, how to extract keywords from the document, which learning algorithm use for classifier construction. Chapter 3 will be devoted to multiword approach we proposed as the solution for the feature extraction problem and the practical results of this method's application.

2. MACHINE LEARNING IN AUTOMATED TEXT CATEGORIZATION

Intelligent agent is a system that employ Machine Learning or Data-Mining techniques and assist users by finding information and help in Web browsing by retrieving relevant information. Most intelligent agents that use machine Learning techniques actually use the content based approach and learn from the content of text documents. This approach based on a comparison of content it has its roots in information retrieval and popular for systems that work on text data – for example, on Web documents or news. Text learning is the application of machine learning techniques to text databases. There is considerable work underway involving learning on text documents that is not necessary related to the Web. We take a look at some of this work through the prism of three questions we found important for using machine learning for text classification: what representation is used for documents, how is the high number of features dealt with, which learning algorithm is used. We summarise in Table C these questions over some related publications to give an idea about state of the art in using supervised learning on text data.

2.1. Text categorization task. Single-label and multi-label categorization

In general, the task of text categorization may be determined as attribution of the document to some pre-defined category. Formally, the task is to determine function that forms the projection of the set of documents on the set of categories, in other words, every document being associated with the category, where the category is just a symbolic label. The attribution of documents to categories realized on the basis of semantics of the documents not on the basis of metadata (e.g. publication date, document type, publication source, etc.) The result of the function could be association of each document to the unique category or to the

set of categories with the value of conformity to each of them calculated as probability. For example, article about the results of the chess match between Garry Kasparov and Deep Blue computer program could obtain both Sport and Computing categories.

2.2. Document representation

The question is how to represent the page, what kind of words can be taken into account and which structure is more reasonable.

2.2.1. Bag-of-words

The most frequently used document representation in information retrieval and text learning is vector representation (or feature vector representation). It is a bag-of-words representation, meaning that all words from the document are taken and now ordering of words or any structure of text is used. In a set of documents each document is represented as a bag-of-words, including all the words that occur in the set of documents.

Original text	Document representation
Figure 1 illustrated an example of such document representation.	*Figure 1 illustrated an example of such document representation.*

Figure 1. Simple bag-of-words document representation

There is another illustration of bag-of-words document representation using frequency vector on Figure 2. Each word occurring in a set of documents is mapped into a feature. A document is represented as a vector of features (see Table C, examples 1, 2, 4, 5, 6, 7, 8, 12, 14, 15, 16, 17, 19, 22, 26, 27, 29) where

each feature is assigned a frequency (the number of times it occurs in the document).

Original text	Document representation using frequency vector
There is an illustration of bag-of-words document representation using frequency vector on Figure 2. Each word occurring in a set of documents is mapped into a feature. A document is represented as a vector of features where each feature is assigned a frequency (the number of times it occur in the document)	*6 a* *1 an* *1 as* *1 assigned* *1 bag* *... ...* *1 times* *1 using* *2 vector* *1 where* *1 word* *1 words*

Figure 2. Bag-of-words document representation using word frequency

And some other systems (see Table C, examples 3, 9, 13, 18, 20, 21, 23, 24, 28, 31, 33, 36, 37) that learn on text use the bag-of-word representation using Boolean features indicating if specific word occurred in document instead frequency of word in a document. Figure 3 illustrated such approach.

Original text	Document representation using frequency vector
And some other systems that learn on text use the bag-of-word representation using Boolean features indicating if specific word occurred in document instead frequency of word in a document. Figure 3 illustrated such approach.	*Yes a* *No also* *Yes and* *Yes approach* *No author* *... ...* *Yes using* *No weather* *Yes word*

word in a document. Figure 3 illustrated such approach.	

Figure 3. Bag-of-words document representation using boolean indicators

There is also some work that uses additional information such as word position or word tuples called n-grams (Table C examples 10, 11).

2.2.2. Methods based on HTML structure

Other approaches are based on HTML document structure (Table C, example 7) [1], only words in between <title> and </title>, and words between <hn> and </hn>, where n is integer, are taken as representative of a page and used as a feature-vector, or, in other words, only document title and accented words are taken into account. Also other approach is used that collects all hyperlinks to some homepage and uses the concatenation of all underlined words as a representation for that page.

2.3. Feature selection

2.3.1. Approaches for increasing feature dimension space

Using bag-of-word document representation and ignoring any additional information we could get large amount of feature words in document-representing vector. Many of these words do not play serious role in our research and their presence may seriously lower the system efficiency.

2.3.2. Stop list

One of the frequently used approaches to reduce the number of different words is to remove words that occur in the "stop list" containing common English words like "a", "the", or "with" (Table C, examples 1, 4, 17, 18, 20, 23, 27, 30, 31, 32, 37). Most of the systems that are listed in the table are using it to reduce the dimensionality of the feature vector space.

2.3.3. Stemming

Word stemming algorithm is used to convert the words to their root form and, as a result, to reduce the number of words in feature vector space. For example, "working" and "works" may be replaced by “work” and it causes frequency of this word “work” to grow and lowers the dimensionality of the feature vector of the document. Algorithm is based on removing suffixes and prefixes from the word and comparing the result with the words from the dictionary. As a result, word-stemming algorithm is language-specific. This algorithm is not very popular among other feature vector reducing techniques (Table C, examples 2, 4, 18, 32, 37). For Perl users exists standard module for English and Deutsche languages, realizing stemming algorithm work also known as Porter algorithm. This module can be found within CPAN, the Comprehensive Perl Archive Network of mirrored FTP sites. But using of this algorithm should be combined with using the dictionary of the main word forms, because it is just removing all the word suffixes and prefixes and may cause different words to be reduced to the same form, for example “r-ing” and “r-ate”.

2.3.4. Approaches for feature selections

Many approaches are language-independent and introduce some sort of word scoring measure to select only the best words. According to the [5] the usual way of learning on text defines a feature for each word that occurred in the training documents. Basically, some evaluation function that is applied to a single feature is used. All the features are independently evaluated, a score is assigned to each of them and the features are sorted according to the assigned score. Then, a predefined number of the best features are taken to form the solution feature subset. Experiments described below represent the results of compare of the various feature-scoring measures.

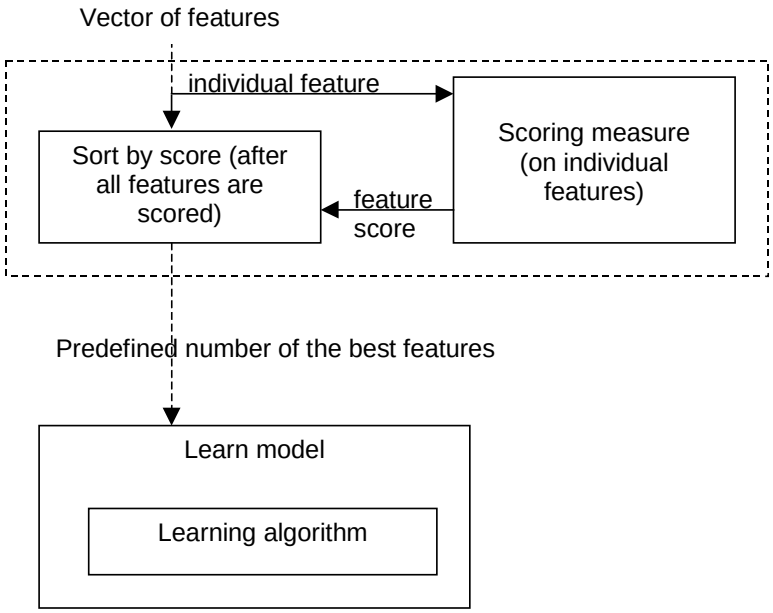


Figure 4: Illustration of the approach to feature subset selection.

One of the questions is how many best features to use in a particular domain and another question is about problem domain observing the influence of the used number of best features to the system performance. The other question is what

scoring (evaluation function) to use since learning on text data has some specifics that should be taken into account when selecting feature subset.

2.3.5. Feature selection experiments

Various number of experiments was performed [4] to appreciate the result of feature subset selection method. They represented that if the words were scored using Information gain many features with high score were not very useful for classification problem, namely, information gain assigns higher score to the features that make better distinction between class values. Other experiments had been proved that the best results are achieved using Odds ratio. All experiments used data with high dimensional space.

In [3] all features were ranked according to their (binary) information gain. Then a naive Bayes classifier was trained using only those features ranked 1-200, 201-500, 501-1000, 1001-2000, 2001-4000, 4001-9962. Results showed that even features ranked lowest still contain considerable information and are somewhat relevant. A classifier using only those "worst" features has a performance much better than random.

Scoring of individual features in feature subset selection approach used on text data can be performed using some of the measures used in machine learning for feature selection during the learning process [4]. In machine learning, several measures are known appropriate for different kinds of class and feature values. In our case we have binary-valued class (is the document relevant for the target concept?) and binary-valued features (does a feature occur in the document?). Notice that for the purpose of feature subset selection, frequency is replaced by occurrence or non-occurrence of a feature. Let's briefly explain these methods.

2.3.6. Document frequency

Document frequency is the number of documents in which a word occurs. (Examples 1, 2, 14, 15, 23, 27, 30, 37, Table C) In this approach we compute the document frequency for each unique word in the document set and removed from the feature space those words whose document frequency was less than some predetermined threshold. The basic assumption is that rare words are either non-informative for category prediction, or not influential in global performance. In either case removal of rare words reduces the dimensionality of the feature space. Improvement in categorization accuracy is also possible if rare words happen to be noise words.

DF thresholding is the simplest technique for vocabulary reduction. It easily scales to very large corpora, with a computational complexity approximately linear in the number of training documents. However, it is usually considered an ad hoc approach to improve efficiency, not a principled criterion for selecting predictive features. Also, DF is typically not used for aggressive word removal because of a widely received assumption in information retrieval. That is, low DF words are assumed to be relatively informative and therefore should not be removed aggressively.

2.3.7. Information gain

It measures the number of bits of information obtained for category prediction by knowing the presence or absence of a word in a document (System examples 3, 14, 20, 28, 31, 33 Table C). Let $C = (c_1, c_2, c_3 ... c_n)$ denote the set of the categories. The information gain of the word W is defined to be:

$$InformationGain(W) =$$
$$-\sum_i P(C_i)\log P(C_i) + P(W)\sum_i P(C_i|W)\log P(C_i|W) + P(W)\sum_i P(C_i|\overline{W})\log P(C_i|\overline{W})$$

Also the definition of the information gain is usually given using the difference of class entropy in data set before and after the usage of the feature F for splitting the data set into subsets. Entropy is the expected amount of information needed to identify an event.

For reducing feature space dimension for the each unique word computes the information gain, and removes from the feature space those words whose information gain was less than some predefined threshold.

2.3.8. Cross Entropy

Cross entropy for text is used on binary-valued features, where only one feature value is important. An example domain is learning over text documents, where features indicate occurrence of words and only cases when the word occurred are important. We denote that value of feature F =`true' as W , trying to make its meaning intuitive (word sequence W occurred in a document). The difference between Cross Entropy and Information gain is that instead of calculating average over all possible feature values, only the value denoting that word (or in our case word sequence) W occurred in a document is considered. The summation part giving Cross entropy is the same (W denoting one of the feature values is used here instead of F that is used in Information gain formula). Cross entropy is given with the following formula:

$$CrossEntropy(F) = P(W)\sum_i P\left(C_i/W\right)\log\frac{P\left(C_i/W\right)}{P(C_i)}$$

2.3.9. Odds ratio

In information retrieval a commonly used feature scoring is by Odds ratio (System example 27, Table C), where the problem is to rank out documents according to their relevance for the positive class value C_1 using occurrence of different words as features.

$$OddsRatio(F) = \log \frac{odds\left(W/C_1\right)}{odds\left(W/C_2\right)} = \log \frac{P\left(W/C_1\right)\left(1 - P\left(W/C_2\right)\right)}{\left(1 - P\left(W/C_1\right)\right)P\left(W/C_2\right)}$$

Where $P(W/C_i)$ is the conditional probability of word W occurring given the i-th class value. Notice that Odds ratio is not averaged over all feature values, like Information gain, but only the value recording that word occurred is considered.

2.3.10. Mutual information

Mutual information (system examples 8, 16, Table C) is based on mutual information between two variable values known from information theory. Mutual information for text is calculated as mutual information for a fixed feature value (word W occurred) averaged over all class values.

$$MutualInfo(F) = \sum_i P(C_i) \log \frac{P\left(W/C_i\right)}{P(W)}$$

2.3.11. Latent semantic indexing using truncated singular value decomposition

This is also language-independent approach which reduces the dimensionality using Latent Semantic Indexing (LSI) [2] (Example 5, 6, 12, Table C), which tries to overcome the problems of lexical matching by using statistically derived conceptual indices instead of individual words for retrieval. LSI assumes that there is some underlying or latent structure in word usage that is partially obscured by variability in word choice. A truncated singular value decomposition (SVD) is used to estimate the structure in word usage across documents. Retrieval is then performed using the database of singular values and vectors obtained from the truncated SVD. Performance data show that these statistically derived vectors are more robust indicators of meaning than individual words.

2.4. Document classification. Learning algorithms

Learning algorithm is an algorithm that is training the classifier using extracted keywords.

2.4.1. Unsupervised and supervised learning

There are two different kinds of learning - unsupervised and supervised learning. On supervised learning the classifier is constructed from the training data. Training data constitutes set of documents previously classified under set of categories.

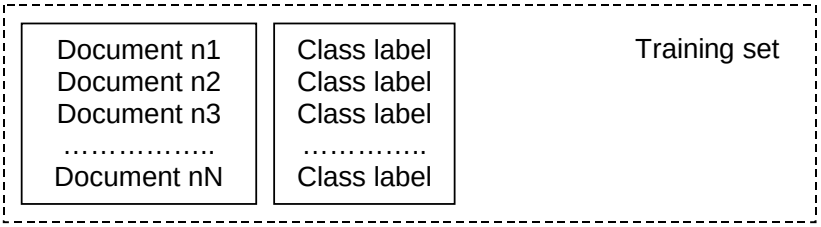


Figure 5. Training set for the supervised learning.

The new classifier based on training data is then tested on fresh test data (test data has same structure as training data) which was not used to train it.

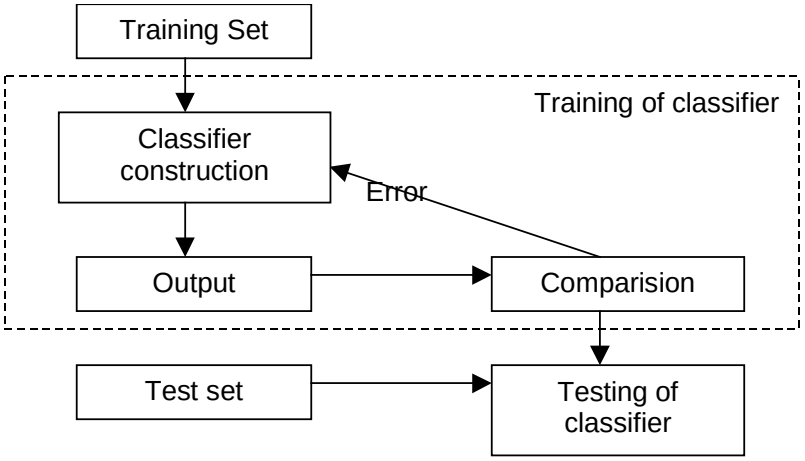


Figure 6. Supervised learning.

Unsupervised learning is the same (generally speaking), but we don't know the classes in the training data.

Figure 7. Training set for the unsupervised learning.

The new classifier is then tested on fresh test data which was not used to train it.

2.4.2. Training set and test set

As previously mentioned, the machine learning approach relies on the existence of a an *initial corpus* $C_o = \{d_1 , \ldots , d_s\}$ of documents previously categorized under the same set of categories $C = \{c_1 , \ldots , c_m\}$ with which the categorizer must operate. For evaluation purposes, in the first stage of classifier construction the initial corpus is typically divided into two sets, not necessarily of equal size:

- a training set $Tr = \{d_1 , \ldots , d_g\}$. This is the set of example documents observing the characteristics of which the classifiers for the various categories are induced;
- a test set $Te = \{d_{g+1} , \ldots , d_s\}$. This set will be used for the purpose of testing the effectiveness of the induced classifiers. Each document in T_e will be fed to the classifiers, and the classifier decisions compared with the expert decisions.

Note that in order to give a scientific character to the experiment the documents in *Te* cannot participate in any way in the inductive construction of the classifiers; if this condition were not to be satisfied, the experimental results obtained would typically be unrealistically good.

2.4.3. TFIDF Classifier

According to the [6] this type of classifier is based on the relevance feedback algorithm originally proposed for the vector space retrieval model (system examples 3, 4, 6, 14, 29, 31, Table C). Due to its heuristic components, there are a number of similar algorithms corresponding to the particular choice of those heuristics. The three main design choices are

- the word weighting method
- the document length normalization
- the similarity measures.

In the following the most popular combination will be presented (known as "tfc"): "tf" word weights, document length normalization using Euclidean vector length and cosine similarity.

Originally developed for information retrieval, the algorithm returns a ranking of documents without providing a threshold to define a decision rule for class membership. Therefore the algorithm has to be adapted to be used for text categorization. The variant presented here seems to be the most straightforward adaptation of the original algorithm to text categorization and domains with more than two categories.

The algorithm builds on the following representation of documents. Each document d is represented as a vector

$$\vec{d} = \left(d^{(1)},\ldots,d^{(|F|)}\right)$$

So that documents with similar content have similar vectors (according to a fixed similarity metric). Each element $d^{(i)}$ represents a distinct word w_i . $d^{(i)}$ for a document d is calculated as a combination of the statistics $TF(w_i, d)$ and $DF(_{wi})$

.The *term frequency TF (w_i,d)* is the number of times word w_i occurs in document *d* and the *document frequency DF (w_i)* is the number of documents in which word w_i occurs at least once. The *inverse document frequency IDF (w_i)* can be calculated from the document frequency.

$$IDF(w_i) = \log\left(\frac{|D|}{DF(w_i)}\right)$$

Here, $|D|$ is the total number of documents. Intuitively, the inverse document frequency of a word is low if it occurs in many documents and is highest if the word occurs in only one. The so-called weight $d^{(i)}$ of word w_i in document *d* is then

$$d(i) = TF(w_i, d) * IDF(w_i)$$

This word weighting heuristic says that a word w_i is an important indexing term for document *d* if it occurs frequently in it (the term frequency is high). On the other hand, words which occur in many documents are rated less important indexing terms due to their low inverse document frequency.

Learning is achieved by combining document vectors into a prototype vector $\vec{c}_j$ for each class C_j. First, both the normalised document vectors of the positive examples for a class as well as those of the negative examples for a class are summed up. The prototype vector is then calculated as a weighted difference of each.

$$\vec{c}_j = \alpha \frac{1}{|C_j|} \sum_{d \in C_j} \frac{\vec{d}}{\|\vec{d}\|} - \beta \frac{1}{|D - C_j|} \sum_{d \in D - C_j} \frac{\vec{d}}{\|\vec{d}\|}$$

α and β are parameters that adjust the relative impact of positive and negative training examples. C_j is the set of training documents assigned to class *j* and $\|\vec{d}\|$ denotes the Euclidean length of a vector $\vec{d}$.

The resulting set of prototype vectors, one vector for each class, represents the learned model. This model can be used to classify a new document d'. Again the document is represented as a vector $\vec{d}'$ using the scheme described above. To classify d' the cosines of the prototype vectors $\vec{c}_j$ with $\vec{d}'$ are calculated. $\vec{d}'$ is assigned to the class with which its document vector has the highest cosine.

$$H_{TFIDF}(d') = \arg\max_{C_j \in C} \cos(\vec{c}_j, \vec{d}')$$

argmax *f(x)* returns the argument *x* for which *f(x)* is maximum and $H_{TFIDF}(d')$ is the category to which the algorithm assigns document d'. The algorithm can be summarised in the following decision rule:

$$H_{TFIDF}(d') = \arg\max_{C_j \in C} \frac{\vec{c}_j}{\|\vec{c}_j\|} * \frac{\vec{d}'}{\|\vec{d}'\|} = \arg\max_{C_j \in C} \frac{\sum_{i=1}^{|F|} c_j^{(i)} * d'^{(i)}}{\sqrt{\sum_{i=1}^{|F|} \left(c_j^{(i)}\right)^2}}$$

In this formula the normalization with the length of the document vector is left out since it does not influence the argmax.

2.4.4. Naive Bayes Classifier

The classifier presented in this section uses a probabilistic model of text. (System examples 7, 8, 20, 23, 26, 27, 31, 33, Table C) Although this model is a strong simplification of the true process by which text is generated, the hope is that it still captures most of the important characteristics.

In the following word based unigram models of text will be used, i.e. words are assumed to occur independently of the other words in the document. There are $|C|$ such models, one for each category. All documents assigned to a particular

category are assumed to be generated according to the model associated with this category.

The following describes one approach to estimating $\Pr(C_j|d')$, the probability that a document d' is in class C_j. Bayes' rule says that to achieve the highest classification accuracy, d' should be assigned to the class for which $\Pr(C_j|d')$ is highest.

$$H_{BAYES}(d') = \arg\max_{C_j \in C} \Pr(C_j|d')$$

$\Pr(C_j|d')$ can be split up by considering documents separately according to their length l.

$$\Pr(C_j|d') = \sum_{l=1}^{\infty} \Pr(C_j|d', l) * \Pr(C_j|d')$$

$\Pr(C_j|d')$ equals one for the length l' of document d' and is zero otherwise. After applying Bayes' theorem to $\Pr(C_j|d', l)$ we can therefore write:

$$\Pr(C_j|d') = \frac{\Pr(C_j|d', l') * \Pr(C_j|l')}{\sum_{C' \in C} \Pr(d'|C', l') * \Pr(C'|l')}$$

$\Pr(d'|C_j, l')$ is the probability of observing document d' in class C_j given its length l'. $\Pr(C_j|l')$ is the prior probability that a document of length l' is in class C_j. In the following we will assume, that the category of a document does not depend on its length, so $\Pr(C_j|l') = \Pr(C_j)$. An estimate $\hat{\Pr}(C_j)$ for $\Pr(C_j)$ can be calculated from the fraction of training documents that is assigned to class C_j.

$$\hat{\Pr}(C_j) = \frac{|C_j|}{\sum_{C' \in C} |C'|} = \frac{|C_j|}{|D|}$$

$|C_j|$ denotes the number of training documents in class Cj and $|D|$ is the total number of documents.

The estimation of $\Pr(d'|Cj, l')$ is more difficult. $\Pr(d'|Cj, l')$ is the probability of observing a document *d′* in class C_j given that we consider only documents of length *l′*. Since there is even for a simplifying representation as used here a huge number of different documents, it is impossible to collect a sufficiently large number of training examples to estimate this probability without prior knowledge or further assumptions. In our case the estimation becomes possible due to the way documents are assumed to be generated. The unigram models introduced above imply that a word's occurrence is only dependent on the class the document comes from, but that it occurs independently of the other words in the document and that it is not dependent on the document length. So $\Pr(d'|Cj, l')$ can be written as:

$$\Pr(d'|Cj, l') \approx \prod_{i=1}^{|d'|} \Pr(w_i|Cj)$$

w_i ranges over the sequence of words in document *d′* which are elements of the feature vector F . $|d'|$ is the number of words in document *d′*. The estimation of $\Pr(d'|Cj)$ is reduced to estimating each $\Pr(w_i|Cj)$ independently. A Bayesian estimate is used for $\Pr(w_i|Cj)$.

$$\hat{\Pr}(w_i|Cj) = \frac{1 + TF(w_i, C_j)}{|F| + \sum_{w' \in |F|} TF(w', C_j)}$$

TF (w,C$_j$) is the overall number of times word *w* occurs within the documents in class C_j . Assumed that the observation of each word is a priori equally likely. Bayesian estimator works well in practice, since it does not falsely estimate probabilities to be zero

The following is the resulting decision rule:

$$H_{BAYES}(d') = \arg\max_{C_j \in C} \frac{\Pr(C_j) * \prod_{i=1}^{|d|'} \Pr(w_i | C_j)}{\sum_{C' \in C} \Pr(C') * \prod_{i=1}^{|d|'} \Pr(w_i | C')} = \arg\max_{C_j \in C} \frac{\Pr(C_j) * \prod_{w \in F}' \Pr(w | C_j)^{TF(w,d')}}{\sum_{C' \in C} \Pr(C') * \prod_{w \in F}' \Pr(w | C')^{TF(w,d')}}$$

If $\Pr(C_j | d')$ is not needed as a measure of confidence, the denominator can be left out since it does not change the argmax.

2.4.5. Decision Tree

According to [7] decision tree text classifier (System examples 2, 20, Table C) consists of a tree in which internal nodes are labelled by terms, branches departing from them are labelled by tests on the weight that the has in the representation of the test document, and leaf nodes are labelled by (not necessarily different) categories. Such a classifier categorizes a test document d_i by recursively testing for the weights that the terms labelling the internal nodes have in the representation of d, until a leaf node is reached; the label of this leaf node is then assigned to d_i. Most such text classifiers assume a binary document representation, and thus consist of binary trees. An example of such a tree is illustrated in Figure 8.

A possible procedure for the induction of a decision tree for category C_i from a set of training examples consists in a "divide and conquer" strategy of recursively

- checking whether all the training examples have the same label (either C_i or $\overline{C}_i$);

• if not, selecting a word *W*, partitioning the training examples into classes of documents that have the same value for *W* , and placing each such class in a separate subtree.

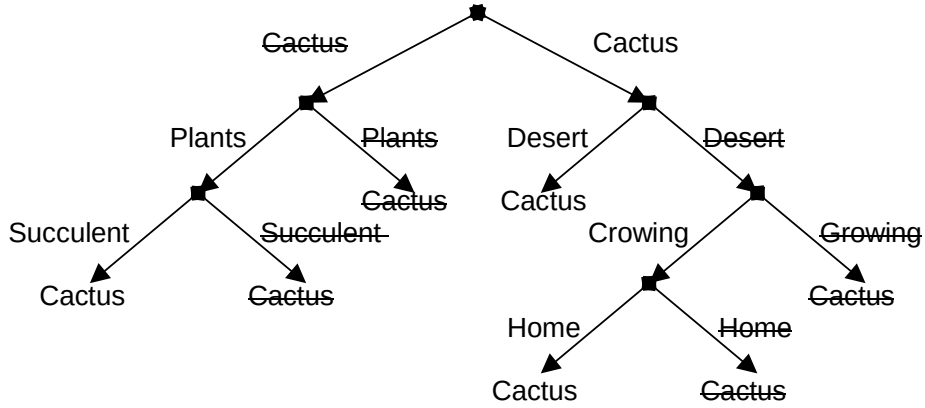


Figure 8. Decision tree. Edges are labelled by words and leaves labelled by categories.

The process is recursively repeated on the subtrees until each leaf node of the tree so generated contain training examples assigned to the same category C_i, which is then chosen as the label for the leaf node. The key step of this process is the choice of the word *W* on which to operate the partition, a choice that is generally made according to an information gain or entropy criterion. However, such a "fully grown" tree may be prone to overfitting, as some branches may be excessively specific to the training data. Any decision tree induction method thus includes a method for growing the tree and one for pruning it, i.e. for removing the overly specific branches so as to minimize the probability of misclassifying test documents.

2.4.6. Decision rules

Such an expert system typically consisted of a set of manually defined rules (one per category) of type if DNF Boolean formula then category, to the effect that if the document satisfied DNF Boolean formula (Table C, examples 1, 10, 11, 28). (DNF standing for "disjunctive normal form "), then it was classified under category. There is an example illustrated in Figure 9.

Cactus & Desert -> **Cactus**
Cactus & Growing & Home -> **Cactus**
Succulent & Plants -> **Cactus**

Figure 9. Classifier for the Cactus category. Keywords occurring in document are indicated in *italic,* categories are indicated in **bold**. DNF rule equivalent to the decision tree on Figure 8

Rule induction methods usually attempt to select from all the possible covering rules (i.e. those rules that correctly classify all the training examples) the "best "one according to some minimality criterion. While decision trees are typically induced through a top-down, "divide-and-conquer "strategy, DNF rules are often induced in a bottom-up fashion. At the start of the induction of the classifier for C_i, every training example is viewed as a clause $w_1 ,...,w_n$ -> γ_i ,where $w_1 ,...,w_n$ are the words contained in the document and γ_i equals C_i or $\overline{C_i}$ according to whether the document is a positive or negative example of C_i .This set of clauses is already a DNF classifier for C_i, but obviously scores tremendously high in words of overfitting. The induction algorithm employs then a process of generalization whereby the rule is simplified through a series of modifications (e.g. removing premises from clauses, or merging clauses) that maximize its compactness while at the same time not affecting the "covering" property of the classifier. At the end of this process, a "pruning "phase similar in spirit to that

employed in decision trees is applied, where the ability to correctly classify all the training examples is traded for more generality.

2.4.7. Neural Network

A neural network classifier is a network of units, where the input units usually represent terms, the output unit(s) represent the category or categories of interest, and the weights on the edges that connect units represent conditional dependence relations. For classifying a test document d_i, its term weights w_{ki} are assigned to the input units; the activation of these units is propagated forward through the network, and the value that the output unit(s) take up as a consequence determines the categorization decision(s). A typical way of training neural networks is backpropagation, where by the term weights of a training document are loaded into the input units, and if a misclassification occurs the error is "backpropagated" so as to change the parameters of the network and eliminate or minimize the error.

2.4.8. K-Nearest Neighbour

The idea of algorithm is that for deciding whether document d_i should be classified under category C_i, k–Nearest Neighbour looks at whether the k training documents most similar to d_i have also been classified under C_i; if the answer is positive for a large enough proportion of them, a positive categorization decision is taken, and a negative decision is taken otherwise (Table C, examples 31, 37).
Lets represent the k-NN method graphically as in Figure 10.

From this figure, it is quite evident that this approach is naturally geared towards document-pivoted categorization. In theory, for category-pivoted-categorization one would only need to "inhibit" the arcs from all training documents to categories $C_1, \ldots C_{i-1}, C_{i+1}, \ldots, C_m$. The resulting network is a classifier for category

C_i only (the original network can instead be seen as running the classifiers for all categories in parallel). One would need to run all documents through this network, one at a time. In practice, though, this would be unacceptably inefficient, since the entire training set would have to be re-ranked m times, one for each category.

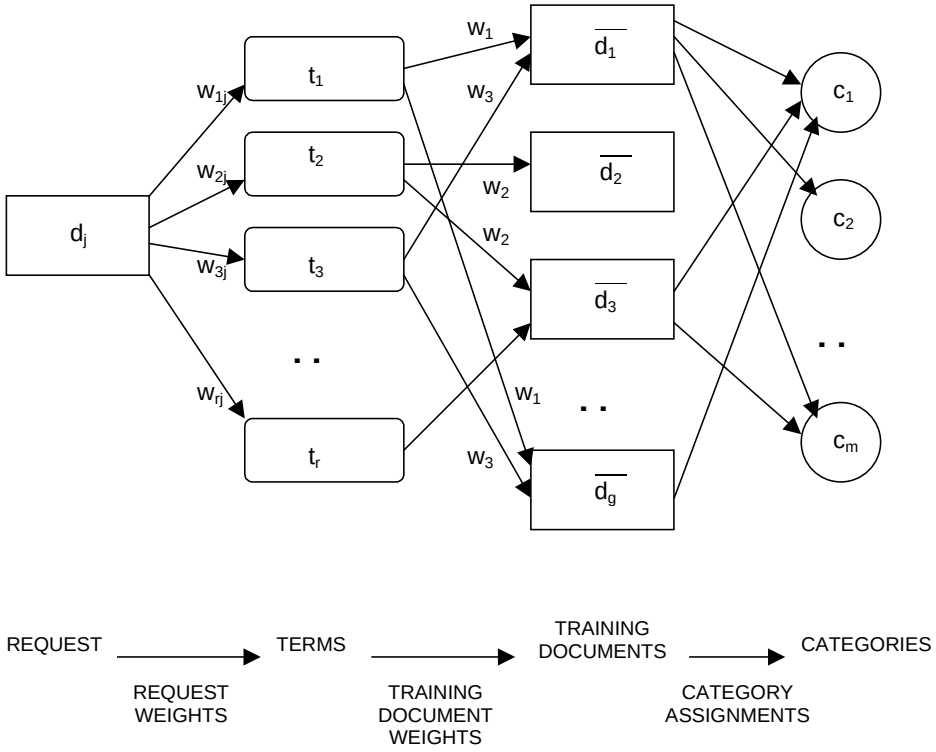


Figure 10. A graphical representation of the k-NN method. Node d_i has weight equal to 1. Weights flow from left to right and get multiplied by the weights of the edges through which they flow; weights incoming into the same node are summed together.

As was mentioned in [8] one of the advantages of k-NN is its efficiency, as the classification of a document in the *m* categories of interest can be performed in

time linear in the cardinality *g* of the training set. Nevertheless, it has to be remarked that “lazy ”learning methods like k-NN are less efficient at classification time, since they do not have a training phase and thus perform all the computation at classification time.

2.4.9. SOM

Let's describe Self-Organizing Map using real known system WEBSOM (another system example 36, Table C). **WEBSOM** is a method for organizing miscellaneous text documents onto meaningful maps for exploration and search. WEBSOM is based on the SOM algorithm that automatically organizes the documents onto a two-dimensional grid so that related documents appear close to each other. The SOM is an algorithm used to visualize and interpret large high-dimensional data sets. Typical applications are visualization of process states or financial results by representing the central dependencies within the data on the map.

The map consists of a regular grid of processing units, "neurons". A model of some multidimensional observation, eventually a vector consisting of features, is associated with each unit. The map attempts to represent all the available observations with optimal accuracy using a restricted set of models. At the same time the models become ordered on the grid so that similar models are close to each other and dissimilar models far from each other.

Fitting of the model vectors is usually carried out by a sequential regression process, where $t = 1,2,...$ is the step index: For each sample $\mathbf{x}(t)$, first the winner index c (best match) is identified by the condition

$$\forall i, \left\| x(t) - m_c(t) \right\| \leq \left\| x(t) - m_i(t) \right\|$$

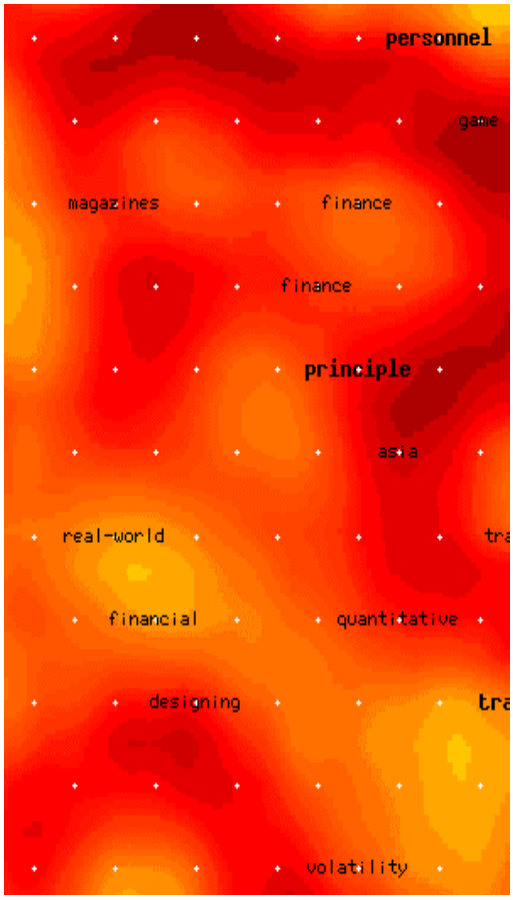
After that, all model vectors or a subset of them that belong to nodes centered around node $c = c(\mathbf{x})$ are updated as

$$m_i(t+1) = m_i(t) + h_{c(x),i}(x(t) - m_i(t))$$

Here $h_{c(x),i}$ is the "neighborhood function", a decreasing function of the distance between the ith and cth nodes on the map grid. This regression is usually reiterated over the available samples**.**

Below the small example of a system view is shown.

WEBSOM zoomed map - comp.ai.neural-nets



Automatically generated labels and examples of titles in which the labels have occurred

asia - looking for job related to NN in HK or Asia region.

designing - Neural nets in designing ATM

finance - PhD on NNs in finance, Where in UK??

financial - Opinion about using ANN in financial analysis?

game - Math functions, stock forecasting & game smarts

magazines - Good NN magazines

personnel - Personnel selection

principle - Looking for Principle Component Analysis Info...

quantitative - Job-Offered: Senior Quantitative Analyst

real-world - FL & NF Applic. to Real-World Problems

volatility - CIFEr'97 Tutorial on Models for Stochastic Volatility - NY, March 23, 1997

With the WEBSOM method a textual document collection may be organized onto a graphical map display that provides an overview of the collection and facilitates interactive browsing. Interesting documents can be located on the map using a content-directed search. Each document is encoded as a histogram of word categories that are formed by the Self-Organizing Map (SOM) algorithm based on the similarities in the contexts of the words. The encoded documents are organized on another Self-Organizing Map, a document map, on which nearby locations contain similar documents. Demo version of the working system may be found on WEBSOM project page (http://websom.hut.fi).

This is a real working system and we can mention that according to the [27] the classification accuracy of this system is about 64%.

2.4.10. Measures of categorization effectiveness

The evaluation of document classifiers is typically conducted experimentally, rather than analytically. The experimental evaluation of classifiers, rather than concentrating on issues of efficiency, usually tries to evaluate the effectiveness of a classifier, i.e. its capability of taking the right categorization decisions.

2.4.11. Precision and recall

Classification effectiveness is most often measured of precision (Pr) and recall (Re), adapted to the case of document categorization. *Precision wrt* C_i (Pr_i) is defined as the probability that if a random document d_x is classified under C_i , this decision is correct. Analogously, *recall wrt* C_i (Re_i) is defined as the probability that, if a random document d_x ought to be classified under C_i , this decision is taken.

These category-relative values may be averaged, in a way to be discussed shortly, to obtain Pr and Re, i.e. values global to the whole category set. Borrowing terminology from logic, Pr may be viewed as the "degree of soundness" of the classifier wrt the given category set C, while Re may be viewed as its "degree of completeness" wrt C.

As they are defined here, Pr_i and Re_i (and consequently *Pr* and *Re*) are to be understood, as subjective probabilities, i.e. values measuring the expectation of the user that the system will behave correctly when classifying a random document under C_i. These probabilities may be estimated in terms of the contingency table for category C_i on a given test set.

Table A. Contingency table.

Category C_i		Expert judgments	
		YES	NO
Classifier	YES	TP_I	FP_i
Judgments	NO	FN_I	TN_I

Here, FP_i (*false positives wrt* C_i , also known as *errors of commission*) is the number of documents of the test set that have been incorrectly classified under C_i ; TN_i (*true negatives wrt* C_i), TP_i (*true positives wrt* C_i) and FN_i (*false negatives wrt* C_i, also known as *errors of commission*) are defined accordingly. Estimates (indicated by carets) of precision wrt C_i and recall wrt C_i may thus be obtained as:

$$\hat{\mathrm{Pr}}_i = \frac{TP_i}{TP_i + FP_i}$$

$$\hat{\mathrm{Re}}_i = \frac{TP_i}{TP_i + FN_i}$$

For obtaining estimates of precision and recall relative t the whole category set, two different methods may be adopted:

- microaveraging: precision and recall are obtained by globally summing ver all individual decisions
- macroaveraging: precision and recall are first evaluated “locally” for each category, and then “globally” by averaging over the results of the different categories.

2.4.12. Which classifier is the best?

In order to compare different classifiers take a notice on a table taken from [10]. Some classifiers mentioned below we haven't referred in our review, but we found it necessary to add them to the comparative table.

Classifier comparison was achieved by using either

- *direct comparison:* classifiers C' and C" are experimented on the same test collection by using a common evaluation measure;
- *indirect comparison:*
  1. classifier C' is tested on test collection 1 and classifier C" is tested on test collection 2;
  2. one or more "baseline"' classifiers C1 , . . . , Cm are tested on both text collection 1 and text collection 2.

Test 2 can give an indication of the relative "hardness" of the two collections; using both this indication and the results from Test 1 yields an indication on the relative value of the two classifiers. The results reported in this evaluation are illustrated in the following table:

Table B. Comparative results among different classifiers obtained on five different document collections. **Bold** indicates the best performer on the collection.

			#1	#2	#3	#4	#5
		# of documents	21,450	14,347	13,272	12,902	12,902
		# of training documents	14,704	10,667	9,610	9,603	9,603
		#of test documents	6,746	3,680	3,662	3,299	3,299
		#of categories	135	93	92	90	10
System	**Type**	**Results reported by**					
Word	(non-learning)	[Yang 1999][10]	.150	.310	.290		
	Probabilistic	[Dumais et al.1998][14]				.752	.815
	Probabilistic	[Joachims 1998][3]					.720
	Probabilistic	[Lam et al.1997][17]	.443				
PropBayes	Probabilistic	[Lewis 1992][18]	.650				
BIM	Probabilistic	[Li and Yamanishi 1999]				.747	
	Probabilistic	[Li and Yamanishi 1999]				.773	
Nb	Probabilistic	[Yang and Liu 1999][8]				.795	
	Decision trees	[Dumais et al.1998][14]					.884
C4.5	Decision trees	[Joachims 1998][3]					.749
Ind	Decision trees	[Lewis and Ringuette 1994][19]	.670				
SWAP-1	Decision rules	[Apte at al. 1994][11]		.805			
Ripper	Decision rules	[Cohen and Singer 1999][12]	.683	.811		.830	
SleepingExperts	Decision rules	[Cohen and Singer 1999][12]	.753	.759		.827	
DL-Esc	Decision rules	[Li and Yamanishi 1999][20]				.820	
Charade	Decision rules	[Moulinier and Ganascia 1996][21]		.738			
Charade	Decision rules	[Moulinier at al. 1996][22]		.783			
LLSF	Regression	[Yang 1999][10]		.855	.810		
LLSF	Regression	[Yang and Liu 1999][8]				.849	
Balanced window	On-line linear	[Dagan at al. 1997][13]	.747	.833			
Windrow-hoff	On-line linear	[Lam and Ho 1998][16]				.822	
Rocchino	Batch linear	[Cohen and Singer 1999][12]	.660	.748		.776	
FindSim	Batch linear	[Dumais et al.1998][14]				.617	.649
Rocchino	Batch linear	[Joachims 1998][3]					.799
Rocchino	Batch linear	[Lam and Ho 1998][16]				.781	
Rocchino	Batch linear	[Li and Yamanishi 1999][20]				.625	
Classi	Neural network	[Ng at al. 1997][23]		.802			
Nnet	Neural network	[Yang and Liu 1999][8]				.838	
	Neural network	[Wiener at al. 1997][26]			**.820**		
Cis-W	Example-based	[Lam and Ho 1998][16]				.860	
k-NN	Example-based	[Joachims 1998][3]					.823
k-NN	Example-based	[Lam and Ho 1998][16]				.820	
k-NN	Example-based	[Yang 1999][10]	.690	.852	**.820**		
k-NN	Example-based	[Yang and Liu 1999][8]				.856	
	SVM	[Dumais et al.1998][14]				.870	**.920**

SvmLight	SVM	[Joachims 1998][3]					.864
SvmLight	SVM	[Li and Yamanishi 1999][20]				.841	
SvmLight	SVM	[Yang and Liu 1999][8]				.859	
Adaboost.MH	Committee	[Shapire and Singer 1999][24]		.860			
	Committee	[Weiss at al. 1999][25]				.878	
	Bayesian net	[Dumais et al.1998][14]				.800	.850
	Bayesian net	[Lam et al.1997][17]	.542				

The published experimental results, and especially those listed in Table B, allow us to attempt some considerations on the comparative performance of the text classification methods discussed. Concerning the relative performance of the classifiers in the work [10] author attempt a few conclusions:

- Boosting-based classifier committees, support vector machines, example-based methods, and regression methods deliver top-notch performance. There seems to be no sufficient evidence to decidedly opt for either method, and it looks that efficiency consideration or application-dependent issues might play a role in breaking such a close tie.
- Neural networks and on-line linear classifiers work very well, although slightly worse than the previously mentioned methods.
- Batch linear classifiers (Rocchio) and probabilistic Naive Bayes classifiers definitely look the worst of learning-based classifiers.
- The data in Table B seem hardly sufficient to say anything about decision tree classifiers.
- By far the lowest performance is displayed by WORD, a “mock” classifier described in [10] and not including any learning component. This unequivocally shows that learning techniques are definitely the way to go for automatic text classification. WORD based on the comparison between documents and category names; each treated as a vector of weighted terms in the vector space model. It was implemented with the only purpose of determining the difference in effectiveness that adding a learning component to a classifier brings about.

It is however important to bear in mind that the considerations above are not absolute and final judgements (if there may be any) on the comparative effectiveness of these text classification methods. One of the reasons is that a particular applicative context may exhibit very different characteristics from the ones to be found in document collection, and different classifiers may respond differently to these characteristics. An experimental study mentioned in [3] involving support vector machines, k -NN, decision trees, Rocchio and Naive Bayes, showed all these classifiers to have a similar effectiveness on categories with >300 positive training examples per category. The fact that this experiment involved the methods which have scored best (support vector machines, k -NN) and worst (Rocchio and Naive Bayes) according to Table B results is indicative of the fact that conditions different from those of documents collection may very well invalidate conclusions drawn on this latter.

2.5. Summary table of algorithms used in some text-learning approaches

In review [9] most of the presented systems were collected and classified by their approaches. The reviewed systems use the context-based approach to the text categorization and learn from the content of the text documents. We add some interesting from our point of view systems to this collection and presenting it here.

Table C. Document representation, feature selection, and learning algorithms used in some text-learning approaches. (The bag-of-words representation is used on Boolean features unless those word frequency us used – frq.)

2.5.1.1.1 Authors	Document representation	Feature selection	Classification
1. C. Apte, F. Damerau, and S.M. Weiss, "Toward Language	Bag of words	Stop list +	Decision

	Independent Automated Learning of Text Categorization Models," *Proc. Seventh Ann. Int'l ACM-SIGIR Conl. Research and Development in Information Retrieval,* ACM Press, New York, 1994, pp. 23 –30.	(frq)	frequency weight	rules
2.	C. Aple, F. Damerait, and S.M. Weiss, "Text Mining with Decision Rules and Decision Trees," *Working Notes of Learning from Text and the Web,* Cont. *Automated Learning and Discovery (COHALD-9S),* Carnegie Melton Univ., Pittsburgh, 1998; ttp://www.cs.emu.edu/'conald/conald.shtml.	Bag of words (frq)	Stemming + min frq	Boosted dicision trees
3.	R. Armstrong et al., "WebWatcher: A Learning Apprentice tor the World Wide Web," *AAAI 1995 Spring Symp. Information Gathering from Heterogeneous, Distributed Environments,* AAAI Press, Menio Park, Calif.. 1995.	Bag of words	Informativity	TFIDF, Winnow, WordStat
4.	M. Balabanovic and Y. Shoham, "Learning Information Retrieval Agents: Exper-ments with Automated Web Browsing," *AAAI 1995 Spring Symp. Informa-tion Gathering from Heterogeneous, Distributed Environments,* AAAI Press, Menlo Park, Calif., 1995	Bag of words (frq)	Stop list + stemming	TFIDF
5.	B.T. Bartell, G.W. Cottrell, and R.K. Belew, "Latent Semantic Indexing is an Optimal Special Case of Multidimensional Scaling," *Proc. ACM SIG Information Retrieval,* ACM Press, New York, 1992, pp. 161-167.	Bag of words (frq)	LSI (Latent sem indexing using SVD	-
6.	M.W. Berry, S.T. Dumais, and G.W. O'Brein, "Using Linear Algebra for Intelligent Information Retrieval," *SIAM Review,* Vol. 37, No. 4, Dec. 1995, pp. 573-595.	Bag of words (frq)	LSI	TFIDF
7.	Y. Q. Choon, Classification of World Wide Web Documents, http://www.cs.emu.edu/'conald/conald.shtml, 1997	Bag of words (frq)	Headers, titles from HTML	Naïve Bayes
8.	M.Craven, D. DisPaquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, S. Slattery, Learning to Extract Symbolic Knowledge from the World Wide Web, http://www.cs.cmu.edu/~conald/conald.shtml, 1998	Bag of words (frq)	Mutual info	Naïve Bayes
9.	R.M. Creecy et al., "Trading MIPS and Memory for Knowledge Eng.," *Comm. ACM,* Vol. 35, Mo. 8, Aug. 1992, pp. 48-64.	Bag of words	-	Mem based reasoning
10.	W.W. Cohen, "Learning to Classify English Text with ILP Methods," *Workshopon Inductive Logic Programming,* CS Dept., K.U. Leiiven, 1995, pp. 3-24.	Bag of words + word position	Minimum frq	Decision rules ILP
11.	W.W. Cohen and Y. Singer, "Context-sensitive Learning Methods for Text Categorization," Proc. *19th Ann. Int'l ACM SIGIR Conl. Research and Development in Information Retrieval (SIGIR '96),* ACM Press, New York, 1996,pp. 307-315.	Ordered word list	-	Decision rules sleeping expert
12.	V. Dasigi, Information Fusion Experiments for the Text	Bag of words	LSI	

Classification, 1998 IEEE	(frq)		
13. B. Geifand, M. Wulfekuhler, and W.F. Punch til, "Automated Concept Extraction from Plain Text," *Working Notes! !of Learning from Text and the Web, Conf. Automated Learning and Discovery (COIVALD-98),* Carnegie Mellon Univ., Pittsburgh, 1998, http://www.cs.cmu.edu/~conald/conald.shtml.	Bag of words + WordNet	Minimum connectivity	Semantic relationship graph
14. T.A. Joachims, "Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization," *Proc. 14th Int'l Conl. Machine Learning (ICML97),* Morgan Kaufmann, San Francisco, 1997, pp. 143-151.	Bag of words (frq)	Minimum frq +informativiy	TFIDF, PrTFIDF, naïve Bayes
15. T.A. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," *Proc. 10th European Conf. Machine Learning (ECML 98),* Springer-Verlag, Berlin, 1998, pp. 137-142.	Bag of words (frq)	Minimum frq	Support Vector Machines
16. W. Lam, K.F. Low, and C.Y. Ho, "Using Bayesian Network Induction Approach for Text Categorization," *15th Int'l Joint Conf. Artificial Intelligence (IJCA197),*AAAI Press, Menio Park, Calif.. 1997, pp. 745-750.	Bag of words (frq)	Mutual info	Bayesian network
17. W. Lam and C.Y. Ho, "Using A Genaralized Instance Set for Automatic Text Categorization," *Proc. 21th Ann. Int'l ACM SIGIR Conl. Research and Development in Information Retrieval (SIGIR '98),* ACM Press, New York, 1998,pp.81-89.	Bag of words (frq)	Stop list	Generalized instance set k-nearest neighbor
18. W.Lam, K. Low, Automatic Document representation based on Probabilistic Reasoning: Model and Perfomance analysis, 1997 IEEE	Bag od words	Stop list + stemming	
19. S.Lam, D.Lee, Feature reduction Neural Network based Text Categorization, 1999 IEEE	Bag of words (frq)	DF+PCA+CF -DF+TFIDF	NN
20. D.D. Lewis and M. Ringuette, "Comparison of Two Learning Algorithms for Text Categorization," *Proc. Third Ann. Symp. Document Analysis and Information Retrieval,* Information Sciences Research Inst., Las Vegas, 1994, pp. 81-93.	Bag of words	Stop list + informativity	Naïve bayes, decisn trees
21. D.D. Lewis and W.A. Gale, "A Sequential Algorithm for Training Text Classifiers," *Proc. Seventh Ann. Int'l ACM-SIGIR Conf. Research ana Development in Information Retrieval,* ACM Press. New York, 1994.	Bag of words	Log likelihood ratio	Logistic regression with nbayes
22. D.D. Lewis etal., "Training Algorithms for Linear Text Classifiers," *Proc. 1Sth Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '96),* ACM Press, New York, 1996, pp. 298-306.	Bag of words (frq)	-	Widrow-Hoff, EG
23. J.Li, A.Jain, "Classification of Text Document" , 1998 IEEE .	Bag of words	Stoplist + min freq	Naïve Bayes
24. R. Liere and P. Tadepalli, "Active Learning with Committees:	Bag of words	-	Winnow (in

Preliminary Results in Comparing Winnow and Perceptron in Text Categorization," *Working Notes* or *Learning from Text and the Web, Conf. Automated Learning and Discovery ICONALD-98),* Carnegie Mellon Univ., Pittsburgh, 1998, http://www.cs.cmu.edu/-conald/conald.shtml.			query by committee)
25. P. Maes, "Agents that Reduce Work and Information Overload," *Comm. ACM.* Vol. 37, No. 7, July 1994. pp. 30-40.	Bag of words+ header info	Selecting keqwords	Mem-based reasoning
26. D. MIadenic, *Personal WebWatcher: Implementation and Design,* Tech. Report IJS-DP-7472, Carnegie Melton Univ., Pittsburgh, 1996, http://www.cs.cmu.edu/~TextLearnJng/pww	Bag of words (frq)	Informativity	Naive bayes nearest neig
27. D. MIadenic and M. Grobelnik, "Feature Selection for Classification Based on Text Hierarchy, *Working Notes of Learning from Text and the Web, Conf. Automated Learning and (CONALD-98),* Carnegie Mellon Univ., Pittsburgh, 1998.	Bag of words using n-grams (freq)	Stop list + Minimum frq + odds ratio	Naïve bayes
28. Moulinier and J.-G. Ganascia, "Applying an Existing Machine Learning Algorithm to Text Categorization," *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing,* S. Wermter, E. Riloff, and G. Scheter, eds., Springer-Verlag, Berlin, 1996, pp. 343-354.	Bag of words	Informativity	Decision rules
29. A. Mouradi, Text Categorization using the Semi-supervised Fuzzy c-Means Algorithm, 1999 IEEE	Bag of words (frq)	Stoplist, weight word	TFIDF
30. K. Nigam and A. McCallum, "Pool-Based Active Learning for Text Classification." *Working Notes of Learning from Text and the Web, Conf. Automated Learning and Discovery (CONALD-98),* Carnegie Mellon Univ., Pittsburgh, 1998; http://www.cs.cmu.edu/~conald/conald.shtml	Bag of words	Minimum frq	EM with QBC
31. M. Pazzani and D. Bitlsus, "Learning and Revising User Profiles: The Identification of Interesting Web Sites," *Machine Learning 27,* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997. pp. 313-331.	Bag of words	Stop list + Informativity	TFIDF,k-N N, N bayes, NN, decis tr
32. J. Shavlik and T. Eliassi-Rad, "Building Intelligent Agents for Web-Based Tasks: A Theory-Refinement Approach," *Working Notes of Learning from Text and the Web, Conf. Automated Learning and Discovery (CONALD'98),* Carnegie Mellon Univ., Pittsburgh, 1998, http://www.cs.cmu.edu/-conald/conald.shtml.	Localized bag of words	Stop list + Stemming	Theory refinement on NN
33. S. Slattery and M. Craven, "Learning to Exploit Document Relationships and Structure: The Case for Relational Learning on the Web," *Working Notes of Learning from Text and the Web, Conf. Automated Learning and Discovery (CONALD-98),* Carnegie Mellon Univ., Pittsburgh, 1998, http://www.cs.cmu.edu/-conald/conald.shtml.	Bag of words + hypertext/graph	Informativity	Naïve Bayes ILP

34. M. Mc Elligott and H, Sorensen, "An Emergent Approach to Information Filtering," *Abakus. U.C.C. Computer Science J.,* Vol. 1. No. 4. Dec. 1993, pp. 1-19. 35. H. Sorensen and M. McElligott, "PSUN: A Profiling System for Usenet News." *CIKM'95 Intelligent Information Agents Workshop,* 1995.	n-gram praph (only bigrams)	Weighting graph edges	Connectionist combined with genetic algorithms
36. A. Visa, J.Toivonen, R.Rukonen, B.Back, H.Vanharanta, "Knowledge discavery from Text Documents Based on Paragraph Maps"	Bag of words	Smart encoding	SOM
37. E. Wiener, J.O. Pedersen, and A.S. Weigend, "A Neural Network Approach to Topic Spotting," *Proc. Fourth Ann. Symp. Document Analysis and Information Retrieval (SDAIR '95),* Information Science Research Inst, Las Vegas, 1995; http://www.stern.nyu.edu/~aweigend/Research/Papers/TextCategorization.	Bag of words	Stoplist +Min frq + stemming + relevancy or LSI Informativity c^2 stat	Neural networks, logistic regression k-nearest neighbor, LLSF

3. MULTIWORD APPROACH TO THE FEATURE EXTRACTION TASK

3.1. Motivation

Having made the review of existing methods and approaches to the decision of features allocation problem, knowing the state of the art and using our longtime expertise with hierarchical deterministic classification we have thought up the new multiword approach to the decision of this task that will be stated in this chapter below. For classifier construction we assume to use method SunX. This already developed expert system includes both learning and classification and described in Section 3.8.1. We also used other classifier construction system offered by Matlab and other additional LDA (linear discriminant analysis) libraries for classifier construction for the purpose of testing of our feature extraction method.

3.2. Approach description

The formal description of multiword approach includes the following steps:

1. Document cleaning, i.e. removing of all short, long, and general words, numbers, special characters etc.
2. Specifying the maximal "length of interest" (LI) number, expressing the largest possible number of words (NW), taken into account. Another constant required is the maximal length (in characters) of all elements, extracted from the document applying LI parameter. In practice NW gives the dimension of a feature space.
3. Breaking the whole document into single elements (training samples) based on LI number (if LI=1, the number of samples is equal to the number of words in the original document and the dimensionality is equal

to the longest acceptable word). All readable character are substituted by their ASCII values.

4. To all recently originated vectors the proper output class, corresponding with the article type, is assigned.
5. Performing steps 1-4 for all training documents, checking and removing the duplicate cases (the same patterns cannot be classified to the different classes). This results in the final training (or testing) set, which can be submitted to any standard classifier.

In occasion of using SunX or Matlab classifier the result should be converted to the corresponding format.

3.3. Representation of documents

We made use of simple bag-of-words document representation using frequency vector as described in Section 2.2.1. Each word occurring in a set of documents is mapped into a feature. A document is represented as a vector of features where each feature is assigned a frequency (the number of times it occurs in the document).

3.4. Feature selection

In order to reduce the number of different words we removed words that occur in the "stop list" containing common English words like "a", "about", "accordingly" etc. In our opinion these words contain insignificant part of the document's information but still appreciably increase the dimensionality of the feature vectors and reduce system performance. For the basis stoplist we took "eign" dictionary file commonly distributed in /usr/share/dict with FreeBSD, this file contains common English stop list with 255 entries. Notice that only with FreeBSD this

file is distributed in as text. Later we extended this file with numerals in their different forms, and designations of date. As a result the file size rised up to 474 entries. All documents in our system were preprocessed in compliance with the stoplist file content and all the words that occur in stoplist were deleted, all numbers and special characters were deleted as well.

Because of the specific of our approach we built classifier using words of the nearly same length, thereto we deleted all words more than 11 symbols or less than 5 symbols long. Words more or less long than mentioned seriously change the dimensionalities of the feature vectors and we decided to get rid of them.

We calculated the document frequency for each unique word in the document set and removed from the feature space those words whose document frequency was less than 3. Document frequency is the number of documents in which a word occurs. The basic assumption is that rare words are neither informative for category prediction, nor influential in global performance. In either case removal of rare words reduces the dimensionality of the feature space. Improvement in categorization accuracy is also possible if rare words happen to be noise words.

Then we calculated different values of “length of interest” (LI) number, expressing the number of words, taken into account. We split up each document into single elements (training samples) basing on LI number. Each group of words (training sample) represents one feature vector. We tested different LI and adduced the results of testing in Section 3.10.3.

At the last step we united all training samples for each category into a separate class file. For each training sample we calculated its occurrence frequency (normalized by number of samples in document and by number of documents in category) for this category using the following formula:

$$SF = \frac{DF(t_i)}{TF(t_i, d) * |D|}$$

where SF – sample frequency , $TF(t_i, d)$ - number of samples t_i which occurs in the document d, $|D|$ - number of documents in this category. $DF(t_i)$ is the number of documents in which sample t_i occurs at least once. We united all class files into one general training set of samples. If equal samples existed in different categories (different class files), we removed sample with the lowest frequency. Notice that SF – sample frequency calculation is identical to weight of term calculation that was described in section 2.4.3. about TFIDF classifier.

3.5. Document classification

For text classification we used two classifiers: SunX and additional modules realizing classifier of linear discriminant analysis for Matlab program package. These classifiers are going to be described in detail in section 3.8.

In order to use offered SunX classifier we represented each training sample by its ASCII values instead of readable characters. We calculated average length of the single word and also the sigma deviation (equal to a square root from dispersion, where dispersion is squared distance between variety supervisions and it's central tendency). Then, we calculated the criterion equal to the average length plus 3 sigma and compared each word in each training sample with criterion. For every word that was longer than criterion, all superfluous symbols were deleted to make its length equal to criterion. For every word that was shorter than criterion its length was supplemented till criterion by "zeros". All data processing, used to reduce word to the suitable form, was performed at the step of feature extraction before the moment of sample frequency count. Its evident that some additional

coincident samples may appear after processing of training samples by the offered method. All such samples should be properly processed on SF - sample frequency count.

3.6. Modules description

We have realized text processing using Perl programming language on Unix platform. Perl seems to us to be the most suitable language for the text processing solutions.

Below we describe actions of the modules that belong to our program.

Each module has directory name for its in parameter that points the directory where to process data, and so the data from the training documents directory and test data directory are processed.

Reduct.pl – used to delete stoplist words, numerals, numbers and all designation of data – weekday and month names, all the words having frequency less than 4 and all the special characters from the whole set of documents and to convert all words into lower case.

Split.pl – for the each document - split the whole document into single elements (training samples) based on LI (length of interest) number. Converts readable characters in each training sample to their ASCII values.

Extract.pl - Calculates average length of every word and also calculates sigma deviation. Calculates the criterion equal to the average length plus 3 sigma. Compares each word in each training sample with criterion and if the word is longer than criterion, deletes all superfluous symbols to make its length equal to

criterion. If the word is shorter than criterion than supplements its length till criterion by "zeros".

Merge.pl – Counts *SF* – sample frequency for each sample. Merges all samples of one class into single file.

Merge_classes.pl – Merges all resulting files into one single file. Resulting file contains samples and category indexes for each sample.

To process resulting data obtained from SunX and lda-classifier by Matlab the following modules were used (modules represent data using classifier's own format):

Rcopy.pl – for SunX classifier, copies data samples to the SunX classifier format. Output file has extension ".pat".

Matlab.pl – for Matlab classifier, copies data samples to the suitable Matlab format.

3.7. Data sets description

We tested the proposed method on small collection of test documents collected from Internet. About a half of them were collected form CNN news site and the rest part was taken from the different Internet news conferences. The reason of using different sources was to represent each category by articles written in official language as well as in spoken language. For this propose we found suitable to take the materials from CNN news and discussion boards where people talk in a natural manner.

Our collection consists of four categories: Sport, Politics, Business and Computing. Every category contains about 250 documents of various sizes. All documents are preprocessed and in our work we deal only with ASCII text pages (exclude background and other web properties). For the better performance all documents in each category were numerated and every document has it's own number and category number in its name and ".txt" extension.

For supervised learning documents were split in two parts: training set and test set. Training set was used to built classifier and after that the classifier was tested on test set. The description of the obtained experimental data is adduced in a table below.

4 categories:

Table D. Description of training and testing sets

	The amount of documents in each category				***The amount of words in dictionary***	
	#1 *Finance*	#2 *Computing*	#3 *Politics*	#4 *Sport*	***Before processing***	***After processing***
Training set	123	171	167	178	8544	7246
Test set	27	37	36	38		

3.8. Learning methods

For text data classifying we used two different approaches in order to compare the results of different classifier creation methods employment..

3.8.1. SunX

The algorithm that we had used on classifier designing purposes provides good results on image classification based on long feature vectors. Therefore we decided to use this approach to build the classifier for our task. Description of the algorithm may be found in [15]. We also have found suitable to describe it here.

Employed classifier generates a piecewise linear discrimination function to separate n real output classes (n=4 in our application). Such approach is applied to the original feature space and preserves consequently real problem background. The learning process includes two main phases.

During the first one, an unsupervised learning principle is applied to divide the normalized feature space into the corresponding number *w* of working classes $(w \geq n)$. Relation between w and n is given by the parameter *Similarity Level* (SL), which defines the initial estimation of uniformity of a working class region. Using, e.g., the Euclidean metric, $SL \in \langle 0,1 \rangle$ and it represents the radius of circle *A* surrounding the currently processed training sample x_A . For each x_A located inside *A* and belonging to the same output class *n* the etalon x_E is calculated and conjoint samples are removed. Such procedure is repeated until all samples are replaced with etalons. Note that the value of quantity w can vary from *n* for *SL=1* to the cardinality of the training set *C* for *SL=0*. Then all etalons are recursively separated and the final structure is stored as a binary tree. Euclidean metric can be used again as a distance measure.

In the second phase, the topology of resultant tree structure (dendrogram) is verified and the exact form of discrimination functions is searched. Each internal tree node represents one etalon and is realized with single standard neuron. The

appropriate subset of real training samples x_I determined in the previous step is dichotomized with the anti-gradient algorithm, the validity of each division line is limited with its predecessors from upper nodes towards to the root. Computational realization of learning procedure ensures standard shape of error function and allows to break the adaptation if the linear separation fails in any node. In such a case, the *SL* value must be decreased and the learning process has to start with a new pre-clustering. Otherwise leaf nodes are recursively pruned (if $w_{LEAF_1} = w_{LEAF_2}$ for the same parent), renumbered $(w \to n)$ and the algorithm is successfully finished.

Input parameters for the classifier :
Similarity level – described above (range [0, 1]).
Error level – permissible classifier error on training data (range [0, 1]).
Learning rate – classifier training rate (range [0, 1]), "the higher, the sooner"
Momentum - (range [0, 1]) term is used in Backpropagation to accelerate convergence. This term is used to avoid oscillations around a particular minimum.

3.8.2. Linear Discriminant Analysis (LDA). Matlab-lda classifier

We developed Matlab program that realises train and test phase using existing Matlab libraries with Linear Discriminant Analysis. Linear Discriminant Analysis (LDA) is common methods used for multi-group data classification. LDA is a transform-based method that attempts to minimize the ratio of within-class scatter to the between-class scatter, thereby maximizing class separability. Classifier construction in Linear Discriminant Analysis uses the following scheme:

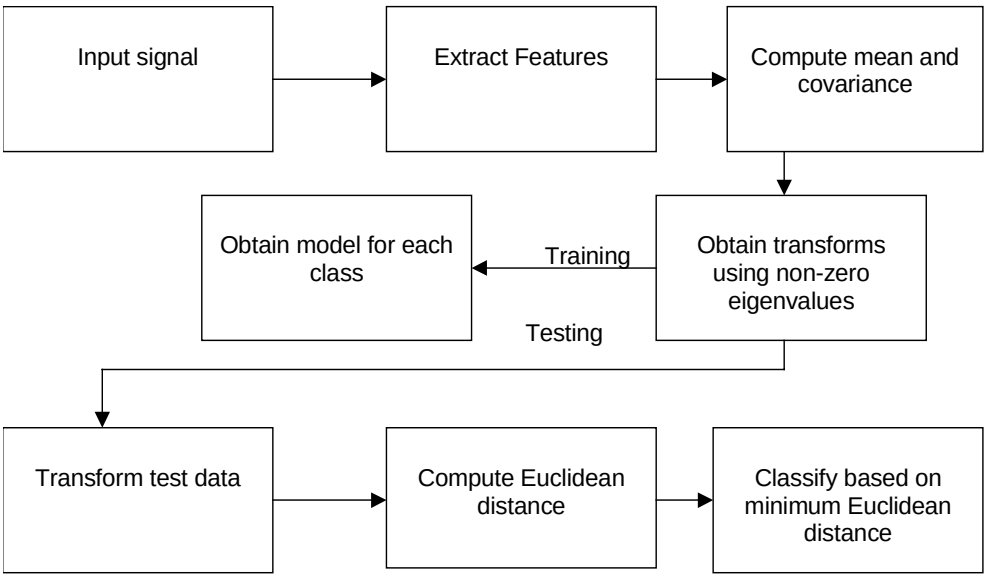


Figure 11. LDA Algorithm.

3.9. Testing results

Testing results obtained on 3 categories

[1] Computing (163 articles)

[2] Politics (167 articles)

[3] Sport (180 articles)

l – number of letters in word taken into account, according to the Section 3.2. If number of letters in the word is less than l then we substitute "zeros" to the end of the word

LI – "length of interest" number of words in word sequence ("window") that is taken as feature vector, number of words in feature vector (LI*l- number of features in feature vector)

step – "step" used for feature extraction in the document (moving "window" "steps" in the document).

Table E. Test results. 3 classes.

method	LI	l	step	Patterns in training	Patterns in test	% of correct results for category			% total average accuracy
						#1	#2	#3	
Sunx	3	7	1	3758	941	33	36	41	37
Lda	3	7	1	37598	9399	37	32	53	41
Lda	7	7	1	57569	14392	41	38	56	45
Lda	7	7	7	8861	2215	42	33	49	42

Results obtained on testing of 4 categories

[1] Finance (150 articles)

[2] Computing (208 articles)

[3] Politics (203 articles)

[4] Sport (216 articles)

Table F. Test results. 4 classes. SunX classifier.

method	LI	l	step	Patterns in training	Patterns in test	Sunx parameters				% total average accuracy
						Similarity level	Error level	Learning rate	Momentum	
Sunx	20	5	1	4563	2478	0	0.4	1	0	30
Sunx	20	5	1	11173	36875	0	0.4	0.9	0	27
Sunx	20	5	1	4563	2478	0	0.2	1	0	30
Sunx	2	5	1	7381	1328	0	0.4	1	0	30

Sunx	10	5	1	5969	2573	0	0.4	1	0	25
Sunx	5	5	1	1753	7821	0.1	0.15	0.7	0.1	26

Table G. Test results. 4 classes. LDA classifier.

method	LI	l	step	Patterns in training	Patterns in test	% of correct results for category				% total average accuracy
						#1	#2	#3	#4	
Lda	5	5	1	12963	4385	39	27	17	53	33
Lda	30	5	1	39278	4083	46	41	15	67	40
Lda	50	5	1	14744	3829	43	47	17	65	39
Lda	***1***	7	1	724	703	29	24	34	0.52	21
Lda	***2***	7	1	14059	3912	34	26	13	47	27
Lda	***3***	7	1	16254	4302	35	27	10	48	27
Lda	***5***	7	1	16737	4385	39	27	17	53	33
Lda	***10***	7	1	16718	4344	38	34	15	56	34
Lda	10	7	***2***	16230	4344	40	35	14	56	34
Lda	10	7	***3***	16637	4344	39	34	14	56	34
Lda	10	7	***7***	16663	4344	38	33	14	55	33
Lda	10	7	***10***	16836	4344	36	33	14	54	32
Lda	10	7	1	***167186***	4344	40	35	12	58	34
Lda	10	7	1	***55728***	4344	38	33	14	56	33
Lda	10	7	1	***33437***	4344	39	34	13	58	34
Lda	10	7	1	***11082***	4344	36	36	14	55	34
Lda	10	7	1	***8359***	4344	37	34	11	23	23
Lda	10	7	1	***4776***	4344	35	32	10	24	22

Lda	10	7	1	***2131***	4344	30	30	16	23	23
Lda	10	7	1	***1001***	4344	32	33	11	22	22
Lda	***15***	7	1	16547	4280	39	39	14	59	35
Lda	***20***	7	1	16327	4305	42	39	14	59	36
Lda	***25***	7	1	16081	4149	38	36	16	62	36
Lda	***50***	7	1	14740	3829	40	42	16	62	36
Lda	***100***	7	1	14433	5471	36	36	14	64	36
Lda	***150***	7	1	10242	6965	38	39	18	66	36
Lda	***200***	7	1	1446	1007	26	25	22	37	27

3.10. Results discussion

3.10.1. Requirements discussion for SunX and LDA Matlab methods

Test results on SunX classifier and Matlab LDA classifier showed that in our environment (SunOS 5.6 Generic_105181-20 sun4u sparc SUNW, Ultra-1) the use of SunX classifier is unproductive as it takes quite a lot of time.

For example attempt to construct SunX classifier on the following data

Table H. Data description. 4 classes. SunX classifier.

LI	l	step	Patterns in training	Sunx parameters			
				Similarity level	Error level	Learning rate	Momentum
30	5	1	15384	0	0.4	1	0

was unsuccessfull, after 2 days work classifier didn't start to calculate first node and still was making the etalons by using similarity level (see algorithm

description above in Section 3.8.2.). We decided to stop training the classifier using this data.

For another data set (described in table below) SunX classifier spent 1,5 day's and successfully had constructed the classifier.

Table I. Data description. 4 classes. SunX classifier.

LI	l	step	Patterns in training	Sunx parameters			
				Similarity level	Error level	Learning rate	Momentum
20	5	1	11173	0	0.4	0.9	0

In attempt to increase parameters (reduce Error level to 0.2) we have got unsatisfactory result. After the few days work classifier was still making the etalons by using similarity level. We decided to interrupt it's work.

We can conclude that classification with SunX works only on the small training sets and spends a lot of time for the classifier training. SunX classifier works good and quick if dimension of feature space is smaller than 100 (product of LI and l) and number of training samples does not exceed ~15000. On this data with classification parameters Similarity level: 0, Error level: 0,4 , Learning rate: 1, Momentum: 0 – classifier training will not exceed 1,5 days.

Classification with Matlab classifier provides better time results, and learning part usually takes not more than half of an hour. For this reason we decided to use Matlab classifier in most of our tests. Another reason is that SunX classifier

doesn't provide for our tests better results than Matlab classifier and this aspect we discuss below.

By experimental testing we have got the best parameters for classification with Matlab LDA classifier. It works if number of training samples does not exceed 500000 and feature dimension space does not exceed 500. Otherwise we may run into a memory lack.

In following sections we compare classifiers on various parameters. We are using the accuracy for this comparison that define percent of "true" results on classifier testing. For the criterion of comparison we use the accuracy, defined as a percent of “true” results of classifier testing.

3.10.2. Comparison of classifiers

The classifiers comparison has been done on 3 categories of documents
Table J. Test results. 3 classes.

method	LI	l	step	Patterns in training	Patterns in test	% of correct results for category			% total average accuracy
						#1	#2	#3	
Sunx	3	7	1	3758	941	33	36	41	37.09
Lda	3	7	1	37598	9399	37	32	53	41.26

Notice that such difference in sizes of training sets does not influence on accuracy results (will be considered below, see Section 3.10.4.).

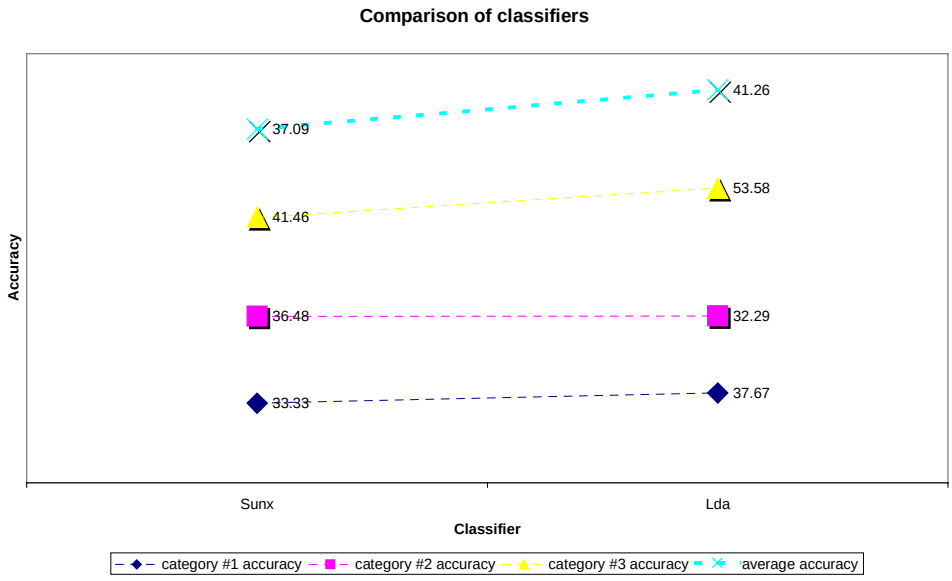


Figure 12. Comparison of classifiers.

Classifiers comparison has shown that with default (average) learning parameters on the same training and test data this classifiers provide quite similar classification accuracy for each class and average result. Notice that SunX learning process takes more time than Matlab LDA.

3.10.3. Analysis of influence of feature extracting parameters to the accuracy

Analysis of influence of feature extracting parameters to the accuracy based on Matlab classifier performance had shown that classifier accuracy increase if we change step of feature extraction.

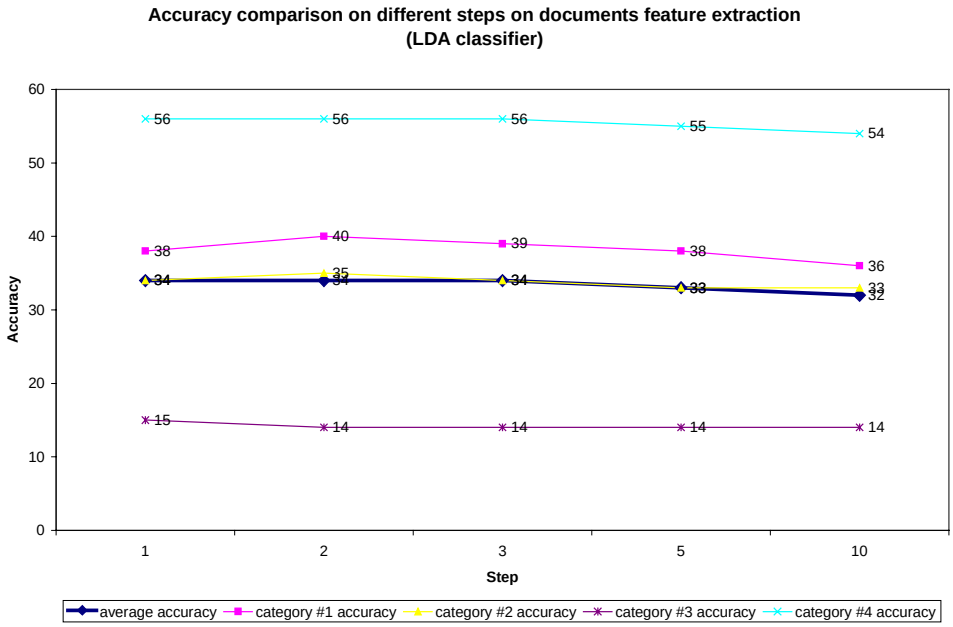


Figure 13. Accuracy comparison on different steps on documents feature extraction (LDA classifier)

This graph is based on the results from the table and it shows that when we change step from 1 to 10 we decrease accuracy for each class and also decrease average accuracy. In this tests for reducing feature space dimension (it is clear that when we use step 1 dimension is 10 times more than dimension with step 10) we take not all but approximately 16000 random samples from training set.

Table K. Test results. 4 classes. LDA classifier.

method	LI	l	step	Patterns in training	Patterns in test	% of correct results for category				% total average accuracy
						#1	#2	#3	#4	

Lda	10	7	***1***	16718	4344	38	34	15	56	34
Lda	10	7	***2***	16230	4344	40	35	14	56	34
Lda	10	7	***3***	16637	4344	39	34	14	56	34
Lda	10	7	***7***	16663	4344	38	33	14	55	33
Lda	10	7	***10***	16836	4344	36	33	14	54	32

Testing have been done on the typical data with Matlab LDA classifier.

We have not done many tests with 3 categories, there are only 2 tests with different step. Following table and graph presented results of these tests.

Table L. Test results. 3 classes. LDA classifier.

method	LI	l	step	Patterns in training	Patterns in test	% of correct results for category			% total average accuracy
						#1	#2	#3	
Lda	7	7	1	57569	14392	41	38	56	45.41
Lda	7	7	7	8861	2215	42	33	49	42.08

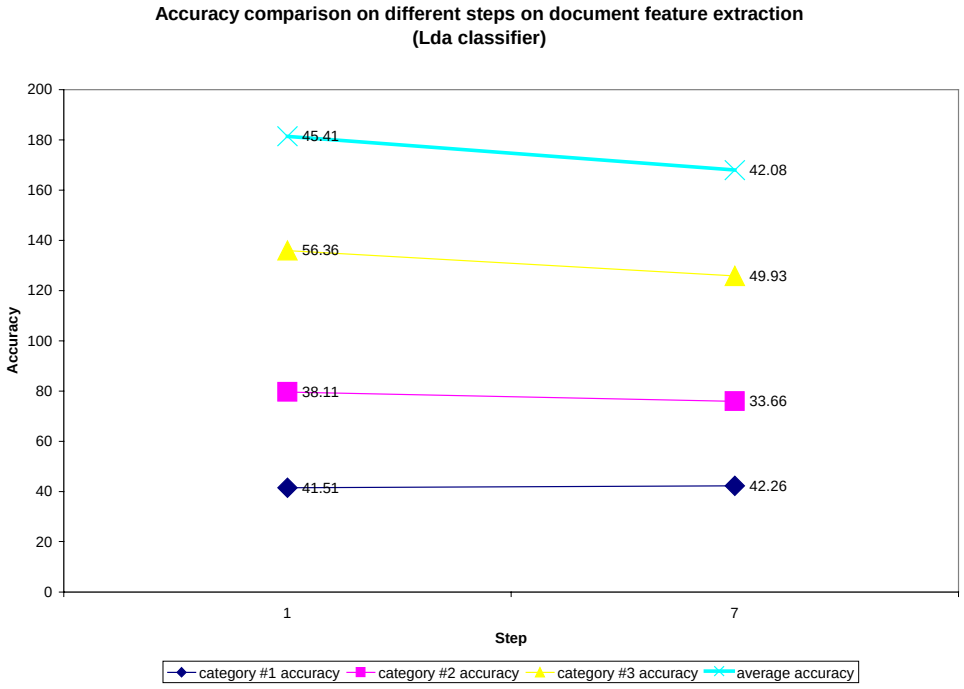


Figure 14. Accuracy comparison on different steps on document feature extraction (Lda classifier)

In general for longer step in feature extracting we got lower result in accuracy.

Also we tested both classifiers using different length of word sequences in feature vectors. For SunX classifier (4 categories) we obtained following results:

Table M. Test results. 4 classes. SunX classifier.

Method	LI	l	step	Patterns in training	Pat-terns in test	Sunx parameters				% total average accuracy
						Similarity level	Error level	Learn rate	Momentum	

Sunx	***2***	5	1	7381	1328	0	0.4	1	0	30
Sunx	***5***	5	1	1753	7821	0.1	0.15	0.7	0.1	26
Sunx	***10***	5	1	5969	2573	0	0.4	1	0	25
Sunx	***20***	5	1	4563	2478	0	0.4	1	0	30

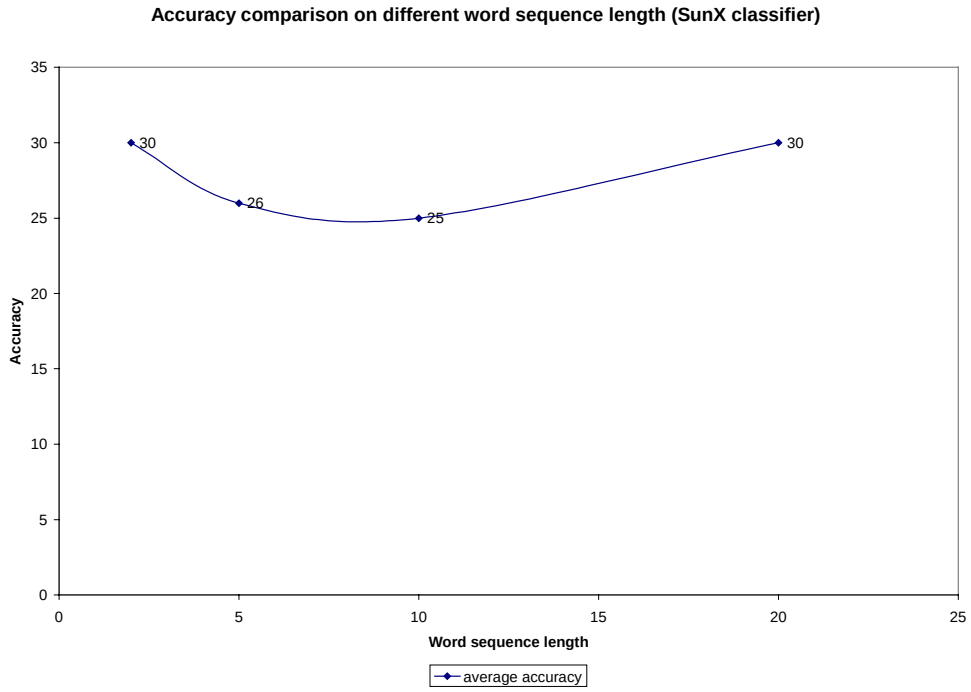


Figure 15. Accuracy comparison on different word sequence length (SunX classifier).

LDA classifier provides bit different results but the same level than the previous:

Table N. Test results. 4 classes. LDA classifier.

method	LI	l	step	Patterns in training	Pat-terns in test	% of correct results for category				% total average accuracy
						#1	#2	#3	#4	
Lda	***1***	7	1	724	703	29	24	34	0.52	21
Lda	***2***	7	1	14059	3912	34	26	13	47	27
Lda	***3***	7	1	16254	4302	35	27	10	48	27
Lda	***5***	7	1	16737	4385	39	27	17	53	33
Lda	***10***	7	1	16718	4344	38	34	15	56	34
Lda	***15***	7	1	16547	4280	39	39	14	59	35
Lda	***20***	7	1	16327	4305	42	39	14	59	36
Lda	***25***	7	1	16081	4149	38	36	16	62	36
Lda	***50***	7	1	14740	3829	40	42	16	62	36
Lda	***100***	7	1	14433	5471	36	36	14	64	36
Lda	***150***	7	1	10242	6965	38	39	18	66	36
Lda	***200***	7	1	1446	1007	26	25	22	37	27

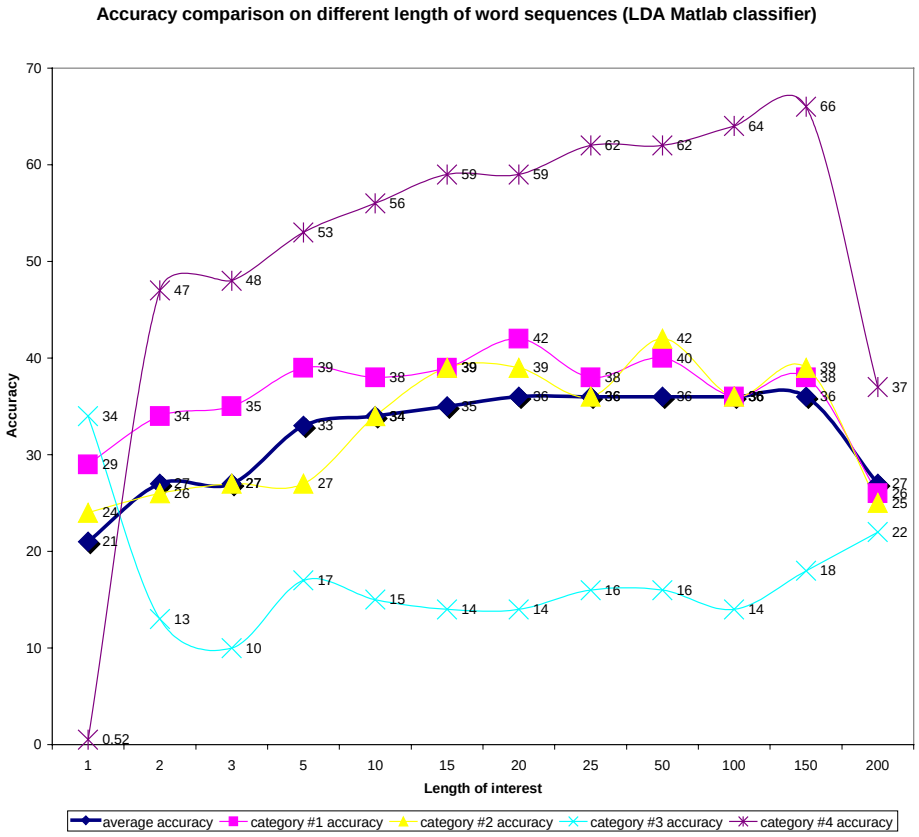


Figure 16. Accuracy comparison on different length of word sequences (LDA Matlab classifier).

In both cases for SunX classifier as well as for Matlbab LDA classifier we found out that for typical (meaning) parameter (5-50 words) values these classifiers provide very similar result accuracy, about 30-35%.

3.10.4. Analysis of cardinality of the training set and accuracy

We decided to test learning ability of Matlab LDA classifier in depending of training set cardinality, for this task we trained classifiers on different sizes of training set and compared resulting accuracy.

Table O. Test results. 4 classes. LDA classifier.

method	LI	l	step	Patterns in training	Patterns in test	% of correct results for category				% total average accuracy
						#1	#2	#3	#4	
Lda	10	7	1	***167186***	4344	40	35	12	58	34
Lda	10	7	1	***55728***	4344	38	33	14	56	33
Lda	10	7	1	***33437***	4344	39	34	13	58	34
Lda	10	7	1	***11082***	4344	36	36	14	55	34
Lda	10	7	1	***8359***	4344	37	34	11	23	23
Lda	10	7	1	***4776***	4344	35	32	10	24	22
Lda	10	7	1	***2131***	4344	30	30	16	23	23
Lda	10	7	1	***1001***	4344	32	33	11	22	22

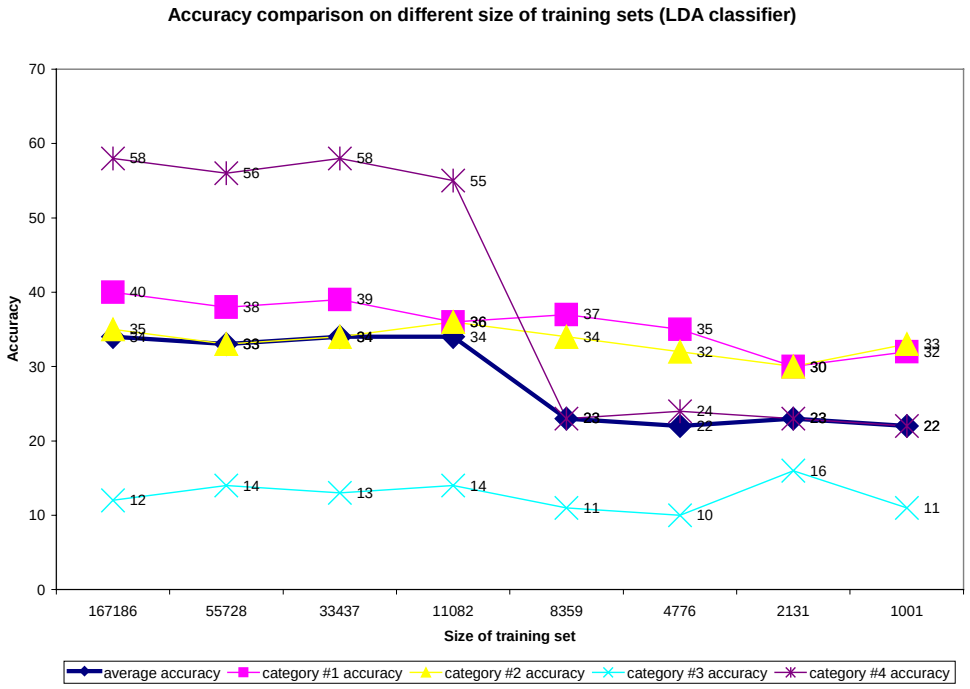


Figure 17. Accuracy comparison on different size of training sets (LDA classifier).

Results of these tests have shown that on training sets with more than 10000 samples Matlab LDA classifier construction provided similar, about 33-35% accuracy of classifier.

3.10.5. The best obtained test case

Unfortunately there are no kind of "best" test case, because in all Matlab tests we obtained quite similar results for average accuracy. The highest average accuracy is about 36 % and its parameters and data description are presented in Table N.

3.10.6. Comparison experiments with 3 and 4 classes of articles.

Comparison of these two kinds of experiments has been done based on the result table mentioned above in Section 3.9.

For SunX classifier we can mention that the average accuracy for all tests which have been done on 3 classes is about 37%. That is 4% more than results which can be obtained during random allocation. Total accuracy for all tests having been done on 4 classes is about 28% that is 3% more than results, which can be obtained during random allocation. We can conclude that on 3 classes classifier works better than on 4 classes. It is not opposite to the common sense.
For LDA Matlab classifier we can dealt with more result data and can compare classification with better accuracy than SunX classification. The meaning accuracy on the 3 classes is 43% that is 10% more than result of random allocation, and for 4 classes the meaning value of accuracy is 33,5% that is 8,5% more than can be during random allocation. So, LDA Matlab classifier also works better on 3 classes than on 4.

4. CONCLUSIONS AND FURTHER WORK.

In this thesis we discussed possible approaches to the task of automated text classification, made a survey of existing systems and present some comparison. Having reviewed existing approaches we developed and implemented own multiword approach to the task of feature extraction and machine learning on text sources. We compared multiword approach to the feature subset selection using two different systems such as linear discriminant analysis based classifier, and classifier combining unsupervised learning for etalon extraction with supervised learning using common back propagation algorithm for neural networks. Test results have shown that effectiveness of multiword approach for 3 categories is about 40-45% and for 4 categories - 35%. As one of the methods for results improvement we recommend to combine our approach with methods using scoring measure; another good idea is to apply dictionary and try to convert all words in feature vectors to their root forms, also it would be better to increase source data space because we are training the classifier using very small documents data set.

5. REFERENCES

[1] Choon, Y. Q. 1997. Classification of World Wide Web Documents, http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/ wwkb/ choon-thesis. html

[2] Berry Berry, M., Dumais, S., Letsche, T. 1995. Computational Methods for Intelligent Information Access, http://www.cs.utk. edu/ ~berry/sc95/sc95. html

[3] Joachims, T. 1998. Text Categorization with support Vector Machines: Learning with many relevant features, Proceedings of ECML-98, 10th European Conference on Machine Learning, Lecture Notes in Computer Science, Number 1398, Springer Verlag, Heidelberg, DE, pp. 137-142, http://www-ai.informatik.uni-dortmund.de/PERSONAL/joachims. html

[4] Mladenic, D. 1998. Feature subset selection in text-learning, Proceedings of ECML-98, 10th European Conference on Machine Learning, Lecture Notes in Computer Science, Number 1398, Springer Verlag, Heidelberg, DE, pp. 95-100, http://www-ai.ijs. si/ DunjaMladenic/home. html

[5] John, C. H., Kohavi, R., Pfeger, K. 1994. Irrelevant Features and the Subset Selection Problem. Proceeding of the 11th International Conference on Machine Learning ICML94, pp. 121-129

[6] Joachims T. 1997. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization**,** Proceedings of ICML-97, 14th International Conference on Machine Learning, Morgan Kaufmann Publishers, San Francisco, US, pp. 143-151, http://www-ai. informatik.uni-dortmund.de/PERSONAL/joachims. html

[7] Mitchell, T. M. 1996. Machine Learning. McGraw Hill, New York, US.

[8] Yang, Y., Liu, X. 1999. A re-examination of text categorization methods Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval, ACM Press, New York, US, pp. 42-49

[9] Mladenic, D. 1999. Text learning and related intelligent agents: a survey, IEEE Intelligent Systems, 14(4), pp. 44-54

[10] Yang, Y. 1996. An evaluation of statistical approaches to MEDLINE indexing, Proceedings of AMIA-96, Fall Symposium of the American Medical Informatics Association, Hanley and Belfus, pp. 358-362

[11] Apt . e, C., Damerau, F. J., Weiss, S. M. 1994. Automated learning of decision rules for text categorization. ACM Transactions on Information Systems 12 , 3, pp. 233 –251.

[12] Cohen, W. W. and Singer, Y. 1999. Context-sensitive learning methods for text categorization. ACM Transactions on Information Systems 17 , 2, pp. 141 –173.

[13] Dagan, I., Karov, Y., Roth, D. 1997. Mistake-driven learning in text categorization. In Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing, Providence, US, pp. 55 – 63.

[14] Dumais, S. T., Platt, J., Heckerman, D., Sahami, M. 1998. Inductive learning algorithms and representations for text categorization. In Proceedings of CIKM-98, 7th ACM International Conference on Information and Knowledge Management, Washington, US, pp. 148 –155.

[15] Voracek, J. 1999. Intelligent real-time control of molding mixtures composition in foundries, In Proceedings of the "IEEE Midnight-Sun Workshop on Soft Computing Methods in Industrial Applications -

SMCIA'99", Kuusamo, Finland, http://tuli. cc. lut. fi/~voracek/ publications.html

[16] Lam, W. and Ho, C. Y. 1998. Using a generalized instance set for automatic text categorization. In Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval, Melbourne, AU, pp. 81 –89.

[17] Lam, W., Low, K. F., Ho, C. Y. 1997. Using a Bayesian network induction approach for text categorization. In Proceedings of IJCAI-97, 15th International Joint Conference on Artificial Intelligence, Nagoya, JP, pp. 745 –750.

[18] Lewis, D. D. 1992. An evaluation of phrasal and clustered representations on a text categorization task. In Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval, Kobenhavn, DK, pp. 37 –50.

[19] Lewis, D. D. and Ringuette, M. 1994. A comparison of two learning algorithms for text categorization. In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, US, pp. 81 –93.

[20] Li, H. and Yamanishi, K. 1999. Text classification using ESC-based stochastic decision lists. In Proceedings of CIKM-99, 8th ACM International Conference on Information and Knowledge Management, Kansas City, US, pp. 122 –130.

[21] Moulinier, I. and Ganascia, J. G. 1996. Applying an existing machine learning algorithm to text categorization. In S. Wermter, E. Riloff, and G. Scheler Eds., Connectionist, statistical, and symbolic approaches to learning for natural language processing, Heidelberg, DE, pp. 343 –354. Springer Verlag. Published in the “Lecture Notes for Computer Science ”series, number 1040.

[22] Moulinier, I., Raskinis, G., Ganascia, J. -G. 1996. Text categorization:a symbolic approach. In Proceedings of SDAIR-96, 5th Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, US

[23] Ng, H. T., Goh, W. B., Low, K. L. 1997. Feature selection, perceptron learning, and a usability case study for text categorization. In Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval, Philadelphia, US, pp. 67 –73.

[24] Schapire, R. E., Singer, Y., Singhal, A. 1998. Boosting and Rocchio applied to text filtering. In Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval, Melbourne, AU, pp. 215 –223.

[25] Weiss, S. M., Apte, C., Damerau, F. J., Johnson, D. E., Oles, F. J., Goetz, T., Hampp, T. 1999. Maximizing text-mining performance. IEEE Intelligent Systems 14 , 4, pp. 63 –69.

[26] Wiener, E., Pedersen, J. O., Weigend, A. S. 1995. A neural network approach to topic spotting. In Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, US, pp. 317-332.

[27] Kohonen, T., Kaski, S., Lagus, K., Salojиarvi, J., Honkela, J., Paatero, V., Saarela A. 2000. Self organization of a massive document collection, IEEE Transactions on Neural Networks, Special Issue on Neural Networks for Data Mining and Knowledge Discovery, volume 11, number 3, pp. 574-585.

[28] Yang, Y., Pedersen, J., O. 1997. A comparative study on feature selection in text categorization Proceedings of ICML-97, 14th International Conference on Machine Learning, pp. 412-420, Morgan Kaufmann Publishers, San Francisco, US, 1997.

[29] Schütze, H., Hull, D., A., Pedersen, J., O. 1995. A comparison of classifiers and document representations for the routing problem Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval, ACM Press, New York, US, pp. 229-237

[30] Hall, M., A., Smith, L., A. 1998. Practical Feature Subset Selection for Machine Learning. In Proceedings of the 21st Australasian Computer Science Conference, pp. 181-191.

[31] Yang, Y. 1994. Expert Network: Effective and Efficient Learning form Human Decisions in Text Categorization and Retrieval, In Proceedings Seventh ACM-SiGR Conference Research and Development in Information Retrieval, ACM Press, New York, pp. 13-22.

[32] Yang, Y. 1999. An Evaluation of Statistical Approaches to Text Categorization, Information Retrieval Journal, Vol. 1, Number 1-2, pp. 69-90