Computing and Informatics, Vol. 28, 2009, 861–893

TEXT CATEGORIZATION AND SORTING OF WEB SEARCH RESULTS

Miloš Radovanović, Mirjana Ivanović, Zoran Budimac

Department of Mathematics and Informatics
Faculty of Science, University of Novi Sad
Trg D. Obradovića 4, 21000 Novi Sad, Serbia
e-mail: {radacha, mira, zjb}@dmi.uns.ac.rs

Manuscript received 15 February 2007; revised 16 June 2008
Communicated by Peter Vojtáš

Abstract. With the Internet facing the growing problem of information overload, the large volumes, weak structure and noisiness of Web data make it amenable to the application of machine learning techniques. After providing an overview of several topics in text categorization, including document representation, feature selection, and a choice of classifiers, the paper presents experimental results concerning the performance and effects of different transformations of the bag-of-words document representation and feature selection, on texts extracted from the *dmoz* Open Directory of Web pages. Finally, the paper describes the primary motivation for the experiments: a new meta-search engine CatS which utilizes text categorization to enhance the presentation of search results obtained from a major Web search engine.

Keywords: Machine learning, text categorization, document representation, feature selection, meta-search engines

Mathematics Subject Classification 2000: 68T50, 62H30, 68T30, 94A17

1 INTRODUCTION

The Internet, with its current structure, the services it provides, and the ways it is being used, is facing the ever growing problem of information overload. Together with the sheer increase of available content, more and more domains of human activity are exploiting the benefits of the Web. Further difficulty lies in the fact

that the Web was initially conceived for human consumption. The core of the Web was built around text-based services and protocols, with no facilities to support machine processing of available information. As a result, new standards designed on top of the old ones did little to change the basic ways the Web was being used from the beginning. The human user was (and still is) sitting in front of a computer and handling information by means of a computer program which, no matter how sophisticated, has little or no knowledge about the true nature and meaning of the content being processed.

At the time of writing, the situation on the Web is somewhat chaotic. Great volumes of information are available in unstructured (plain text) or semi-structured form (HTML). In addition, even more information resides as "dark matter", occupying databases based on which Web-pages and other visible content may be dynamically generated. Beside weak structuring and limited availability, Web data is inherently *noisy*, bursting with false information (either intentional or not), irrelevant content (like spam e-mail or, from an individual user's point of view, all Web-pages not suiting his/her interests), grammatical errors, specific abbreviations, and data which may be plainly offensive and/or illegal (e.g. pornography, piracy). Thus, the logical ambition to computerize more human activities connected with the Web is facing serious difficulties.

The above-mentioned properties of Web data – large volumes, weak structure, and noisiness – make it amenable to application of *machine learning* (ML) techniques. ML provides many useful methods for discovering patterns and inferring knowledge from raw or shallowly processed data, such as (hyper)text commonly found on the Web. Machine learning has the means to perform such tasks *automatically*, bringing the goal of computerization of many human activities on the Web one step closer to realization.

This paper will present an angle on the application of machine learning techniques on the Web, motivated by the construction of a meta-search engine powered by text categorization techniques. After an introduction to text categorization in Section 2 and a discussion of related work in Section 3, two rounds of experiments carried out in order to determine the parameters for the implementation of the meta-search system will be described in Sections 4 and 5. The constructed system is outlined in Section 6, and the final section gives concluding remarks and guidelines for future work.

2 TEXT CATEGORIZATION

Text categorization (TC) is the task of automatically sorting a set of documents into *categories* (or *classes*, or *topics*) from a predefined set [29]. Applications of TC include text filtering (e.g. protection from spam e-mail), word sense disambiguation, and categorization of Web pages.

TC techniques require text to be transformed to an adequate representation, which is the subject of Section 2.1. Section 2.2 focuses on feature selection, while

Section 2.3 turns to classification methods themselves – their evaluation and principles of functioning.

2.1 Document Representation

General-purpose ML techniques are usually designed for examples which have a fixed set of symbolic or numeric features. A dataset is then merely a table where the columns represent features (attributes), and the rows are individual examples. Free-flowing or semi-structured text thus needs to be transformed in order to apply an ML algorithm. The most widely used approach is the bag-of-words representation.

The bag-of-words representation. In the bag-of-words (BOW) representation word order is discarded from a document and single words are treated as features. Actually, other things can be used as features (e.g. phrases) hence textual features are referred to as *terms* instead of *words*. Let W be the *dictionary* – the set of all words (terms) that occur at least once in the set of documents D. The BOW representation of document d_j is a vector of weights $\mathbf{w}_j = (w_{1j}, \ldots, w_{|W|j})$. There are many variations of the BOW representation, depending on the weight values. For the simplest *binary representation*, $w_{ij} \in \{0, 1\}$; the weight $w_{ij} = 1$ if the i^{th} word is present in document d_j, otherwise $w_{ij} = 0$. In the *term frequency representation* (*tf*), $w_{ij} = \text{tf}_{ij}$, the frequency of the i^{th} term in the j^{th} document.

Many transformations of term frequencies are used in practice. *Normalization* (*norm*) can be employed to scale the term frequencies, accounting for differences in the lengths of documents. The *logtf* transformation may also be applied to term frequencies, resulting in the representation:

$$\text{logtf}(d_j) = (\log(1 + w_{1j}), \ldots, \log(1 + w_{|W|j})).$$

The *inverse document frequency* (*idf*) transformation yields the representation:

$$\text{idf}(d_j) = (w_{1j} \log(|D|/\text{docfreq}(D, 1)), \ldots, w_{|W|j} \log(|D|/\text{docfreq}(D, |W|))),$$

where $\text{docfreq}(D, i)$ is the number of documents from D the i^{th} term occurs in. It can be used by itself (with binary weights w_{ij}), or with term frequencies to form the popular *tfidf* representation.

Similarity measures. With the knowledge that documents in the BOW representation are vectors, one would be tempted to use the Euclidean distance measure to express the similarity between two documents. However, the Euclidean measure (along with the whole Minkowski family of metrics) are generally not suitable to problems involving documents in the BOW representation. Therefore, the *cosine similarity measure* is usually employed:

$$\text{sim}(d_i, d_j) = \frac{\sum_{k=1}^{|W|} w_{ki} \cdot w_{kj}}{\sqrt{\sum_{k=1}^{|W|} w_{ki}^2} \cdot \sqrt{\sum_{k=1}^{|W|} w_{kj}^2}} \ .$$

2.2 Dimensionality Reduction by Feature Selection

It is clear that even for small document collections the BOW document vector will have a high dimension. This may hinder the application of ML techniques not only by causing space and time inefficiency, but by degrading the performance of learning algorithms which cannot scale to such high dimensions.

The usual steps in preprocessing textual data are elimination of digits and special characters, and removal of words which appear too infrequently and too frequently in the document set with respect to some predefined thresholds. The removal of too frequent words is also often done with regards to a list of stopwords – words like "I", "the", "with", etc. Such words may inhibit ML algorithms because they do not add any useful information to the BOW model (for a majority of applications).

Stemming can be viewed as another technique for dimensionality reduction, transforming all forms of a word to the same stem. Therefore, not only does it reduce the number of features, but it also captures correlations between words by fusing them, that way possibly improving the performance of machine learning. The problem of algorithmically determining the stem of an arbitrary English word is satisfactorily solved for most ML applications, one of the widely used algorithms being the Porter Stemmer. However, for many languages in which words inflect more than in English such solutions are not possible, and this problem may itself be tackled by ML techniques.

It is often necessary to further reduce the number of features. For classification, there are two distinct approaches to this problem: feature *selection*, where the resulting set of features is a subset of the old one, and feature *extraction*, which derives a completely different (but smaller) set of features. One of the most notable feature extraction methods used on text is latent semantic indexing (LSI), which employs the singular value decomposition (SVD) linear algebra technique to decompose the term-document matrix and transform document vectors to a lower dimensional space. At the same time, the method preserves (and makes explicit) the correlations between features. In this paper, however, we will focus on issues regarding feature selection.

The question which feature selection (FS) tries to answer is this: for a given set of n features representing documents, which of its 2^n subsets to choose such that classification performance is optimal?

2.2.1 The Wrapper Approach

The approach where a classifier is used to evaluate feature subsets is called the *wrapper* approach. Unfortunately, since it requires repeated classifier training the approach is prohibitively slow in textual domains due to the high dimensionality of data and is therefore seldom used.

2.2.2 The Filter Approach

The *filter* approach attempts to determine the importance of a feature based on some measure which is relatively simple to compute. In essence, a feature is considered more "important" if it correlates with the class feature (it is relevant), at the same time not correlating with other features (it is not redundant). There are many ways to formalize this notion, including *term frequency*, *information gain*, *gain ratio*, *symmetrical uncertainty*, *chi square* and *relief*.

Term frequency. The number of documents a feature (term) occurs in, called *term frequency* (which we shall denote TFDR to differentiate term frequency as a dimensionality reduction method from term frequency as a feature weighing method in the BOW representation), is a surprisingly effective way to rank features for selection. A variation of this measure is to count all occurrences of a term in the whole document set. Note that the removal of stopwords is an important preprocessing step to take before using TFDR, otherwise many useless features will be retained (unless stopwords are important in the particular application of the classifier).

Information theoretic measures. Several useful feature ranking measures originate from information theory. The number of bits needed to express event x_i which occurs with probability $\mathrm{P}(x_i)$ is called *information*, expressed as $\mathrm{I}(x_i) = -\log_2 \mathrm{P}(x_i)$. The expected value of I for a random variable containing events x_i is *entropy*:

$$\mathrm{H}(X) = \sum_i \mathrm{P}(x_i)\,\mathrm{I}(x_i) = -\sum_i \mathrm{P}(x_i)\log_2 \mathrm{P}(x_i),$$

and the conditional entropy of X after observing Y:

$$\mathrm{H}(X|Y) = -\sum_j \mathrm{P}(y_j)\sum_i \mathrm{P}(x_i|y_j)\log_2 \mathrm{P}(x_i|y_j).$$

The reduction in entropy of X before and after observing Y, i.e. the average amount of information about X contained in Y, is referred to as *expected cross entropy* [19]:

$$\mathrm{CH}(X,Y) = \mathrm{H}(X) - \mathrm{H}(X|Y).$$

If we consider features as random variables, then the expected cross entropy of a fixed class attribute C and attribute A is known as the *information gain* of A:

$$\mathrm{IG}_C(A) = \mathrm{CH}(C,A) = \mathrm{H}(C) - \mathrm{H}(C|A).$$

The probabilities are calculated from a given dataset, thus entropy becomes a measure of (im)purity of the dataset relative to the classification we wish to achieve [18, p. 55]. The usual approach is to rank every feature with regards to the class using the IG criterion and choose the best features.

A possible problem of IG is its bias towards features with more values. This may be fixed by normalizing IG with the entropy of A, yielding *gain ratio*:

$$\mathrm{GR}_C(A) = \mathrm{IG}_C(A)/\mathrm{H}(A).$$

Another approach is used in the *symmetrical uncertainty* measure:

$$\mathrm{SU}_C(A) = 2\mathrm{IG}_C(A)/(\mathrm{H}(C) + \mathrm{H}(A)).$$

More details on these measures can be found in [9] and [19] (p. 42).

Chi square. The χ^2 measure from statistics can also be used for estimating the correlation between features and the class feature. If n is the size of the dataset, for the simplest binary version of BOW, where attribute $A \in \{a_0, a_1\}$, and binary classification ($C \in \{c_0, c_1\}$), the χ^2 metric is

$$\mathrm{CHI}_C(A) = \frac{n[\mathrm{P}(a_0, c_0)\,\mathrm{P}(a_1, c_1) - \mathrm{P}(a_0, c_1)\,\mathrm{P}(a_1, c_0)]^2}{\mathrm{P}(a_0)\,\mathrm{P}(a_1)\,\mathrm{P}(c_0)\,\mathrm{P}(c_1)}.$$

Relief. A different approach to feature ranking is used by the *relief* measure first introduced by Kira and Rendell [13]. Relief takes a random example from the dataset and locates two of its nearest neighbors (with regards to some vector distance metric), one from the positive and one from the negative class and uses the values of their features to update the relevance of each feature. The procedure is repeated a specified number of times, the more the better (sampling *every* example if possible). Relief was later extended to ReliefF (RF), with added support for multi-class and noisy datasets [15].

2.3 Classification

The process of training a classifier can be viewed as algorithmically building a mathematical model for separating examples of different classes, from the evidence given in a dataset. There are many different ways of doing this, which resulted in the development of many kinds of classifiers with different properties. Some classification algorithms can only discern between two different classes, making them *two-class* (or *binary*) classifiers, others are naturally *multi-class*.

This section attempts to present classification algorithms from the viewpoint of their application on text. First, evaluation of classifiers is discussed, introducing several ways to use datasets, followed by an overview of text-specific evaluation measures and a description of several key algorithms.

2.3.1 Classifier Evaluation

There are three different aspects of classifier performance [29]: *training efficiency*, *classification efficiency*, and *correctness of classification*. Training and classification

efficiency are measured in terms of execution speed and memory consumption, and present very important factors in practical applications of classification. Nevertheless, the attention of the research community is dominated by the correctness aspect, which applies to this paper as well. "Classification performance" mentioned in previous sections was mostly referring to correctness, and will continue in the same manner.

The dataset used for classification is usually divided into the *training set*, used to train the classifier, and the *test set* for evaluating classifier performance. The ratio between the training and test sets (the *split*) may depend on the amount of available data, the particular application and many other factors. The usual splits include 2/1, 3/1, 4/1 and 9/1, and sometimes test sets bigger than the training sets are used.

The *cross-validation* technique is often used to evaluate classifiers. The dataset is split into n subsets, with one used as the test set, and the remaining $n-1$ subsets comprising the training set. This allows for n different measurements, and the whole process can be repeated k times, resulting in *k runs of n-fold cross-validation*. To preserve class distribution, *stratification* may be employed to keep the ratio of the number of examples of each class the same in the training and test sets.

Having multiple measurements means that a statistical test, like the *t-test*, can be used to determine whether the performance of two classifiers is *significantly* different. Furthermore, in a setting where multiple classifiers are being compared over multiple datasets, the number of statistically significant *wins* and *losses* can be counted and the subtracted value of *wins–losses* used to rank the classifiers.

Evaluation measures. Accuracy – the percentage of correctly classified examples is a good measure for evaluating classifiers in a wide variety of applications. However, consider a binary classification problem with *imbalanced class distribution*, where examples from the negative class constitute 95 % of the dataset. Then, the *trivial rejector* (i.e. the classifier which assigns all examples to the negative class) has an accuracy of 95 %, but is totally unusable in practice if the positive class is of any importance.

		Predicted class	
		yes	*no*
Actual class	*yes*	True Positive (TP)	False Negative (FN)
	no	False Positive (FP)	True Negative (TN)

Table 1. Outcomes of binary classification

Evaluation measures which originated from information retrieval (IR) are commonly used to evaluate text classifiers. *Precision* is defined as the ratio of the number of relevant documents that were retrieved and the total number of retrieved documents. In terms of outcomes of binary classification summarized in Table 1,

$$\text{precision} = \text{TP}\,/(\text{TP} + \text{FP}).$$

Recall is the ratio between the number of relevant documents retrieved, and the total number of relevant documents:

$$\text{recall} = \text{TP}/(\text{TP}+\text{FN}).$$

For comparison,

$$\text{accuracy} = (\text{TP}+\text{TN})/(\text{TP}+\text{TN}+\text{FP}+\text{FN}).$$

It can be said that precision and recall are on the opposite sides of the spectrum. The *trivial acceptor* has 100 % recall and very low precision, while a classifier which makes only one positive classification (and it happens to be correct) has 100 % precision and very low recall. Therefore, the two measures are often combined to form the *F-measure*:

$$\text{F}_\beta = \frac{(\beta^2+1)\cdot\text{precision}\cdot\text{recall}}{\beta^2\cdot\text{precision}+\text{recall}}\ .$$

When $\beta = 1$, F-measure represents the harmonic mean of precision and recall. For $0 \leq \beta < 1$ precision is given more importance ending with $\text{F}_0 = \text{precision}$, while $\beta > 1$ means recall gets the upper hand with the other extreme at $\text{F}_\infty = \text{recall}$. Besides $\beta = 1$, the usual values of β that are used are $\beta = 0.5$ when precision is considered more important, and $\beta = 2$ when recall is preferred [19].

2.3.2 Algorithms

Practically every existing classifier has been tried on text to date [29], making TC quite a popular benchmark for old and new classification algorithms. It would be infeasible to review them all, therefore a selection of algorithms was made. Hopefully, they do well to illustrate the principles, the diversity of approaches, and the state-of-the-art in the field.

Perceptrons. The *perceptron* is a binary classifier which uses the value of the inner product of vectors $\mathbf{v}\cdot\mathbf{x}$ to classify an instance $\mathbf{x}$ according to the previously learned vector of weights $\mathbf{v}$. If the inner product is greater than or equal to some threshold t, the instance is assigned to the positive class and vice versa. More precisely, for binary class $c \in \{-1, 1\}$, $c = \text{sg}(\mathbf{v}\cdot\mathbf{x} - t)$, where $\text{sg}(x) = 1$ if $x \geq 0$, and $\text{sg}(x) = 0$ otherwise. This means that $\mathbf{v}$ and t define a hyperplane which linearly separates the vector space.

Learning the vector $\mathbf{v}$ starts by assigning it a $\mathbf{0}$ vector (or a vector of small positive weights) and continues by examining each training instance $\mathbf{x}$ one at a time and classifying it using the currently learned $\mathbf{v}$. If the classification is incorrect the vector is updated: $\mathbf{v} \leftarrow \mathbf{v} \pm \eta\mathbf{x}$, where addition (subtraction) is used when $\mathbf{x}$ belongs to the positive (negative) class and η is a small positive number – the *learning rate*. The effect of the update is to shift the weights of $\mathbf{v}$ towards the correct classification of $\mathbf{x}$. The algorithm iterates multiple times over instances in the training set until some stoppage criterion is met.

The perceptron showed solid performance on text [28], despite its simplicity. There are numerous extensions, one of them being the *voted-perceptron* by Freund and Schapire [7].

In the voted-perceptron algorithm, *all* vectors $\mathbf{v}$ calculated during training are retained together with the number of training instances they "survive" without being modified. Then, for a list of such weighed perceptrons $(\mathbf{v}_1, c_1), \ldots, (\mathbf{v}_k, c_k)$, the classification is calculated as the sign of the weighed sum of the classifications of each saved perceptron: $c = \mathrm{sg}\left(\sum_{i=1}^{k} c_i \mathrm{sg}(\mathbf{v}_i \cdot \mathbf{x})\right)$, assuming all thresholds are zero. The voted-perceptron was shown to be effective on high-dimensional data, at the same time being simple to implement and having a low training time [7].

Support vector machines. One of the most sophisticated and best performing classifiers applied on text [29] is the support vector machine (SVM) classifier. It is a binary classifier, and its main idea lies in using a *kernel function* whose main effect is to transform the input vector space. The new vector space usually has a higher number of dimensions, with the transformed data being linearly separable. Quadratic programming methods are applied to find a *maximum margin hyperplane*, i.e. the optimal linear separation in the new space, whose inverse transformation should yield an excellent classifier in the original space.

Although the foundations for SVMs were laid out by Vapnik in the 1970s [30], the computational complexity of various solutions to the quadratic programming problem restricted the use of SVMs in practice. Only relatively recently were approximate solutions derived which enabled feasible and, compared with some other classifiers, superior training times. One solution was by Osuna et al. [22], improved by Joachims [11] and implemented in his SVM^{light} package. An alternative is Platt's *sequential minimal optimization* (SMO) algorithm [23] available, for instance, as part of the WEKA workbench [31].

Support vector machines can handle very high dimensions, making them suitable for application on text data without dimensionality reduction [17].

Naïve Bayes. The probabilistic approach to modeling data yielded several machine learning techniques which can be applied on text. One of the most well known and widely used is the *naïve Bayes classifier*. Let random variable C denote the class feature, and $A_1, A_2, \ldots, A_n$ the components of the attribute vector. Then, the class of document vector $\langle a_1, a_2, \ldots, a_n \rangle$ is $c = \mathrm{argmax}_{c_j \in C} \mathrm{P}(c_j | a_1, a_2, \ldots, a_n)$ (in the case that one class maximizes the expression). The Bayes theorem gives

$$\begin{aligned} c &= \operatorname*{argmax}_{c_j \in C} \frac{\mathrm{P}(a_1, a_2, \ldots, a_n | c_j)\,\mathrm{P}(c_j)}{\mathrm{P}(a_1, a_2, \ldots, a_n)} = \operatorname*{argmax}_{c_j \in C} \mathrm{P}(a_1, a_2, \ldots, a_n | c_j)\,\mathrm{P}(c_j) \\ &= \operatorname*{argmax}_{c_j \in C} \mathrm{P}(c_j) \prod_{i=1}^{n} \mathrm{P}(a_i | c_j)\,. \end{aligned}$$

The last derivation used the assumption that attributes are mutually independent, which obviously does not hold in reality, hence the prefix "naïve". Nevertheless, the

approximation has been shown to work in practice. Training involves approximating the values $\mathrm{P}(c_j)$ and $\mathrm{P}(a_i|c_j)$ from data. Several approaches exist, depending on the assumed data distribution. The approach most often used on text involves the multinomial model, and was recently subjected to several enhancements [27, 12]. In the classification phase, if multiple classes maximize the expression $\mathrm{P}(c_j)\prod_{i=1}^{n}\mathrm{P}(a_i|c_j)$ different strategies may be employed to resolve the ambiguity, e.g. choosing the class with the highest prior probability $\mathrm{P}(c_j)$, or simply choosing one of the maximizing classes randomly.

Nearest neighbor classifiers. The training phase of *nearest neighbor* classifiers is practically trivial and consists of storing all examples in a data structure suitable for their later retrieval. Unlike other classifiers, all computation concerning classification of an unseen example is deferred until the classification phase. Then, k instances most similar to the example in question – its k *nearest neighbors* – are retrieved and the class is computed from the classes of the neighbors. Determining the class can consist of just choosing the majority class of all the neighbors, or distance weighing may be used to reduce the influence of faraway neighbors on the classification decision. The similarity function for textual documents is usually the cosine of the angle between vectors.

Decision trees. A decision tree (DT) is a tree whose internal nodes represent features, where arcs are labeled with outcomes of tests on the value of the feature from which they originate, and leaves denote categories. Classifying a new instance using a decision tree involves starting from the root node and following the branches labeled with the test outcomes which are true for the appropriate feature values of the instance, until a leaf with a class value is reached. One of the widely used decision tree learning algorithms is Quinlan's C4.5 [24]. Learning a DT with C4.5 involves choosing the most informative feature using a combination of the IG and GR criteria, determining how best to split its values using tests, and repeating the process recursively for each branch/test. Recursion stops when the tree perfectly fits the data, or when all feature values have been used up. Pruning is performed on the tree to avoid overfitting.

Decision trees are especially useful when the decision of the classifier needs to be interpreted by humans. As for text, DTs may be unsuitable for many applications since they are known not to efficiently handle great numbers of features; but sometimes they do prove superior, for instance on datasets which contain a few highly discriminative features [8].

3 RELATED WORK

Issues with document representation and feature selection. Although the majority of research in TC employs the bag-of-words approach to document representation [8], studies of the impact of its variations on classification started ap-

pearing relatively recently. Leopold and Kindermann [17] experimented with the SVM classifier with different kernels, term frequency transformations and lemmatization of German. They found that lemmatization usually degraded classification performance, and had the additional downside of great computational complexity, making SVMs capable of avoiding it altogether. A study on the impact of document representation on one-class SVM [32] showed that, with a careful choice of representation, classification performance can reach 95 % of the performance of SVM trained on both positive and negative examples. Kibriya et al. [12] compared the performance of SVM and a variant of the naïve Bayes classifier, emphasizing the importance of *tf* and *idf* transformations for naïve Bayes. Authoritative experimental studies of the impacts of feature selection on text classification include [6] and [34].

Meta-search engines. Generally, meta-search engines aim to improve certain aspects of general-purpose search engines including Web coverage, search result relevance, and presentation. As for general-purpose search engines, information available on meta-search engines is rather sparse, but research has been gaining momentum in recent years. SnakeT is a recent implementation of a meta-search engine which sorts results by clustering Web-page snippets,[1] and also provides Web interfaces for books, news and blog domains [5]. Carrot2 is an open-source "research framework for experimenting with automated querying of various data sources (such as search engines), processing search results and their visualization," which also relies on clustering [21]. Formal concept analysis (FCA) was employed for organizing search results by systems CREDO [3] and, more recently, FooCA [14]. CiiRarchies is a hierarchical clustering engine for Web search results described in [16], while Highlight provides the option to sort results at the outermost level using classification, before resorting to clustering for deeper levels of the topic hierarchy [33]. All these systems provide Web interfaces *and* published results, unlike leading commercial clustering engines (e.g. Vivisimo, KartOO) which keep the details about the employed algorithms in hiding. According to [5], no meta-search engine, research or commercial, has outperformed Vivisimo with regards to the quality of generated clusters, and the majority of engines with known internal workings are rather slow, which limits their usefulness in practice.

A pure classification approach to sorting search results was described in [2], where a closed study involving Internet users of different profiles showed that their category style of presentation was generally preferred over the list model. The authors chose to break up the list of results between categories right at the initial displaying of results, showing only several examples from each category, based on which the user could choose to "expand" a particular topic.

Another way to utilize classification in sorting search results is by means of focused, or *vertical* search: the user is first asked to navigate and fix a particular category of interest, and then post a query [28]. The results would not only be

[1] A Web-page snippet consists of the page's title, link and excerpt.

restricted to the chosen (and related) categories, but also ranked in accordance with that particular line of search, instead of using global link graph analysis.

4 CLASSIFICATION EXPERIMENTS: ROUND ONE

The initial motivation for the experimental work presented in this paper lies in the development a meta-search engine which uses TC to enhance the presentation of search results, described in Section 6. From the context of this system, we intended to answer the three questions posed in [20]:

1. what representation to use in documents,
2. how to deal with the high number of features, and
3. which learning algorithm to use.

In order to keep the number of variables manageable, the experiments were organized in two rounds. The first round focuses on question 1. and its interaction with question 3., trying (but not completely succeeding) to avoid question 2. The second round of experiments, presented in the next section, examines in more detail the issues surrounding feature selection and classification algorithms.

This section provides an outline of the experimental study which examines BOW document representations and their impact on the performance of five classifiers. Using *wins–losses* for evaluation, which was seldom done in TC, enabled the measurement of the effects of different BOW document transformations on each classifier.

Section 4.1 introduces the experimental setup – the datasets, the considered document representations and feature selection methods, and the classifiers. Section 4.2 outlines the most notable results – which representations were found best, and the effects of different transformations: normalization, *logtf* and *idf* on different classifiers.

4.1 The Experimental Setup

All experiments described in this paper were performed using the WEKA machine learning toolkit [31]. The data (documents) are concatenated Web-page titles and descriptions from the *dmoz* Open Directory.[2] Five classical measures were recorded: accuracy, precision, recall, F_1 and F_2. The F_2 measure, which gives emphasis to recall over precision, is included for reasons similar to those in [19, p. 48] where false positives are preferred to false negatives. What this means for categorization of search results is that it is preferred to overpopulate categories to a certain extent over leaving results unclassified in category *Other*.

[2] `www.dmoz.org/`, with the data available in downloadable form at `rdf.dmoz.org/`.

Datasets. For this round of experiments 11 two-class datasets were extracted from *dmoz*: *Arts*, *Business*, *Computers*, *Games*, *Health*, *Home*, *Recreation*, *Science*, *Shopping*, *Society* and *Sports*. Positive examples in each dataset are taken from the corresponding category, while negative examples are extracted in a stratified fashion from the other 10 categories. Stopwords were eliminated. The number of features in the datasets ranges from 3 800 to 4 700, and the number of documents between 620 and 770.

Document representation and feature selection. The bag-of-words representation was used, together with all its transformations described in Section 2.1. For notational convenience, the abbreviations denoting each transformation were appended to the names of datasets, so, for instance, *Arts-norm-tf* refers to the normalized term frequency representation of the *Arts* dataset. All meaningful combinations of the transformations, together with stemming (m), amount to 20 different variations summarized in Table 2. Furthermore, experiences with the *idf* transformation described in Section 4.2 made us consider at least one dimensionality reduction method. We chose a simple method based on term frequencies (denoted TFDR) which eliminates least frequent terms, keeping the total number of features around 1 000. Thus, two variants of datasets were generated – with and without TFDR. All this accounts for a total of $11 \cdot 20 \cdot 2 = 440$ datasets for this round of experiments.

Not stemmed		**Stemmed**	
Not normalized	Normalized	Not normalized	Normalized
01		m-01	
idf		m-idf	
tf	norm-tf	m-tf	m-norm-tf
logtf	norm-logtf	m-logtf	m-norm-logtf
tfidf	norm-tfidf	m-tfidf	m-norm-tfidf
logtfidf	norm-logtfidf	m-logtfidf	m-norm-logtfidf

Table 2. Document representations

Classifiers. The WEKA implementations of classifiers used in this round of experiments are: ComplementNaiveBayes (CNB) – an improved multinomial model naïve Bayes [27, 12], SMO [23], VotedPerceptron (VP) [7], IBk (a variant of k-Nearest Neighbor) [1], and J48 (a variation of C4.5) [24].

It should be noted that we used IBk with $k = 1$ and the Euclidean distance metric. Only in the late phases of experimentation we realized that the Euclidean metric was not suitable to BOW data. In addition, TFDR broke the performance of IBk completely, therefore only results without TFDR are reported for IBk.

Separate experiments were run for each of the five classifiers and each of the 11 categories with the 20 datasets corresponding to document representations. The

reported evaluation measures are averages over five runs of 4-fold cross-validation. For comparison of measures WEKA's corrected resampled t-test was used ($\alpha = 0.05$), and the wins and losses of each document representation summed up over the 11 categories for every classifier.

4.2 Outline of the Results

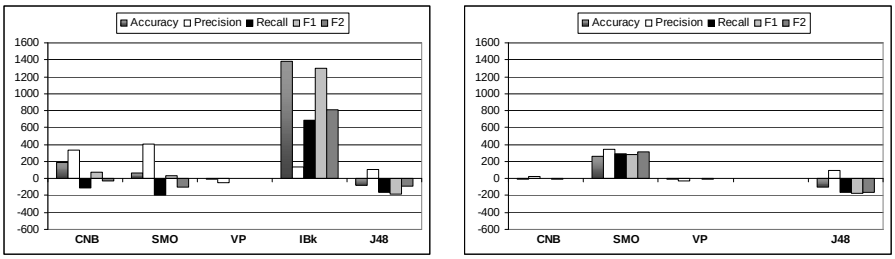
Based on the observed measures of classifier performance and wins–losses summed-up over the datasets, a best representation was identified for each classifier. Table 3 summarizes the performance of classifiers on the *Home* dataset (which best illustrates average classifier behavior over all datasets).

	CNB *m-norm-tf*	**SMO** *m-norm-logtf*	**VP** *m-logtf*	**IBk** *m-norm-logtf*	**J48** *m-logtf*
Accuracy	82.56 (**5.26**)	83.19 (1.67)	78.38 (**5.12**)	74.93 (**21.96**)	71.77 (**3.64**)
Precision	81.24 (**8.66**)	85.67 (**3.86**)	80.45 (**7.85**)	71.32 (**14.32**)	90.24 (1.60)
Recall	83.91 (1.81)	78.93 (3.80)	74.06 (0.96)	81.66 (**45.20**)	47.59 (**10.59**)
F_1	82.48 (**3.64**)	82.07 (2.17)	77.02 (**4.23**)	76.07 (**33.90**)	62.12 (**9.09**)
F_2	83.31 (2.19)	80.14 (3.30)	75.20 (2.16)	79.31 (39.72)	52.48 (**10.41**)

Table 3. Performance of classification (in %) using the best document representations on the *Home* dataset, *without* dimensionality reduction, together with improvements over the worst representations (statistically significant ones are in **boldface**)

The effects of a particular BOW transformation (i.e. stemming, normalization, *logtf* and *idf*) on classification performance were measured by summing-up the wins–losses for datasets with the transformation applied, and again for datasets without the transformation, and examining their difference. These differences may conveniently be depicted on a bar-chart for every classifier and evaluation measure. What follows is a discussion of such results for the *norm*, *logtf* and *idf* transformations.

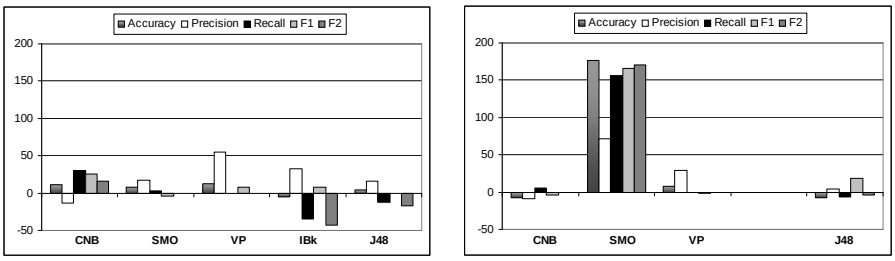
Effects of normalization. The chart in Figure 1 a) suggests that, without TFDR, normalization enhances the performance of CNB and SMO by all measures except recall (and consequently F_2). Biggest improvement was on IBk, virtually no effect was apparent for VP, while the performance of J48 was degraded. The improving effect on SMO can be attributed to the known fact that SMO performs well with small numeric attributes (we disallowed the SMO classifier to employ own data normalization). As for IBk, normalization helped considerably with the comparison of vectors using the Euclidean metric. The worsening of the performance of J48 can be explained by the increased difficulty of finding good numeric intervals within the normalized weights in order to branch the decision tree. After TFDR (Figure 1 b)), CNB joined VP in its insensitivity, while SMO witnessed a big boost of performance when data was normalized.



a) b)

Fig. 1. Effects of *normalization* a) before and b) after TFDR

Effects of *logtf*. The effects of the *logtf* transformation are somewhat different to those of normalization, as can be seen in Figure 2. The effects are mostly those of mild improvement of classification performance. The stronger improvement of SMO in Figure 2 b) can again be attributed to SMO's preference for small numeric values, along with the smoothing effect of *logtf* on the elimination of small weights by TFDR.

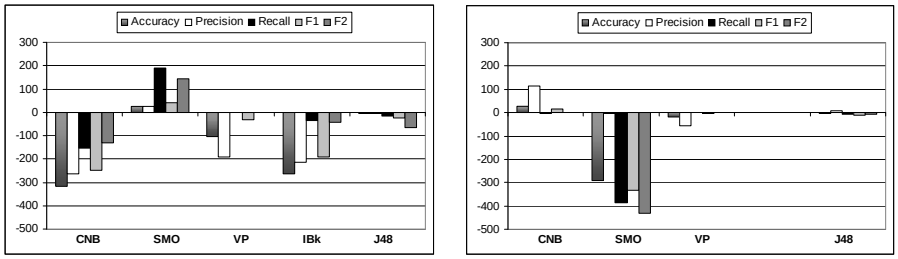


a) b)

Fig. 2. Effects of the *logtf* transformation a) before and b) after TFDR

Effects of *idf*. The *idf* transformation proved to be most interesting regarding the range of effects it had on the classifiers. Without TFDR, all classifiers except SMO are negatively effected by it, as suggested by Figure 3 a). We conjectured that this is due to our data containing many terms which are found only in a small number of documents, that way making *idf* assign them unrealistically high weights. We therefore introduced TFDR, expecting *idf* to improve classification performance on data with infrequent features eliminated. The expected effect did occur, as demonstrated by Figure 3 b), for all classifiers *except* SMO, whose performance drastically degraded! This behavior seemed to be worth investigating further, and was the motivation for the study presented in Section 5.

More details on this round of experiments are given in [26].



a) b)

Fig. 3. Effects of *idf* applied to *tf* a) before and b) after TFDR

5 CLASSIFICATION EXPERIMENTS: ROUND TWO

The *idf* transformation, as depicted in Figure 3, had completely opposite effects on the CNB and SMO classifiers. Without dimensionality reduction (TFDR) the performance of CNB was degraded, while SMO witnessed improvement. With dimensionality reduction, however, the effect was completely reversed. At first glance, this behavior went against our intuition. The *idf* transformation assigns a high score to infrequent terms, and these are roughly the terms which are eliminated by TFDR. Thus, if the less frequent terms are not discriminative (which depends on the data) then the performance of a classifier before dimensionality reduction should be degraded by *idf*, while after dimensionality reduction it is more likely to be improved since TFDR eliminates the features to which *idf* gives an unrealistically high weight. On the other hand, if the infrequent terms are good discriminators between classes, before TFDR *idf* should improve classifier performance, and after TFDR it is more likely to have a degrading effect. But, the two classifiers in question exhibited totally opposing behavior with regards to the above distinction. This raised an interesting question whether the two classifiers are so dissimilar that they were able to model totally different patterns of feature weights and their correspondence to the class.

Together with rankings of feature selection methods and reduction rates, which will be used as a guideline for the implementation of the meta-search system described in Section 6, this section will present a study motivated by the above question. Besides *idf*, we will also examine normalization and *logtf* transformations, i.e. their effect on several commonly used feature selection methods, considering a wide array of reduction rates instead of only two.

Section 5.1 introduces the datasets, document representations, feature selection methods and classifiers used in this round of experiments. Section 5.2 presents the results: the rankings of feature selection methods and reduction rates, and the findings concerning their interactions with the considered document representation transformations.

5.1 The Experimental Setup

Datasets. The subset of *dmoz* corresponding to the 11 top level categories, as described in Section 4, was the basis for the generation of datasets for this study. This time, six larger two-class datasets were extracted, corresponding to top-level topics *Arts*, *Computers* and *Sports*, and second-level topics *Music*, *Roleplaying* and *Physics*. For each dataset, positive examples are taken for the corresponding topic, and negative examples are extracted from all other topics at the same level of the hierarchy within a common parent topic, in a stratified fashion. That is, for the first-level topics, negative examples are taken from the 11-category subset of *dmoz*, while for the second-level categories (*Music*, *Roleplaying* and *Physics*) negative examples are restricted to their first-level parents (*Arts*, *Games* and *Science*, respectively). Stopwords were eliminated, and the Porter Stemmer was applied to every dataset since the best document representations determined in Section 4 all included stemming. The number of terms in the datasets ranges from 10 000 to 15 000, and the number of documents between 5 000 and 9 000.

Document representations. For every dataset, the variations of the BOW representation which were considered best for at least one classifier were generated: *m-norm-tf*, *m-norm-logtf* and *m-logtf*. In order to study the interaction between BOW transformations and feature selection, *m-tf* and *m-norm-tfidf* were also included.

Feature selection. The examined feature selection methods are chi-square, information gain, gain ratio, ReliefF and symmetrical uncertainty. Classification performance was measured on datasets consisting of top 100, 500, 1 000, 3 000, 5 000, 7 000 and 10 000 features selected by each method, and on datasets with all features. The feature selection method which discards least frequent features from Section 4 (TFDR) was not used, since it was not possible to achieve the same reduction rates without randomly discarding features with identical frequencies of appearance.

Classifiers. The same classifiers implemented in WEKA that were used in Section 4 were employed in this study: ComplementNaiveBayes (CNB), SMO, VotedPerceptron (VP), IBk and J48.

Experiments were carried out on each dataset with a particular document representation, FS method and number of features, and classifier. As in Section 4, five runs of 4-fold cross-validation were used, and WEKA's corrected resampled t-test ($\alpha = 0.05$) employed to compare the values of evaluation measures.

5.2 Results

5.2.1 Rankings of Feature Selection Methods and Reduction Rates

Table 4 shows the top five combinations of feature selection methods and reduction rates measured by accuracy, precision, recall, F_1 and F_2, respectively. The wins–

losses (WL) values are summed-up over all datasets, while the actual values of performance measures are averaged.

CNB *m-norm-tf*			**SMO** *m-norm-logtf*			**VP** *m-logtf*			**IBk** *m-norm-logtf*			**J48** *m-logtf*		
FS	WL	acc.	FS	WL	acc.	FS	WL	acc.	FS	WL	acc.	FS	WL	acc.
all	133	**89.4**	chi1000	169	**90.1**	chi1000	144	88.6	gr500	203	84.4	chi500	34	83.3
chi10000	113	89.2	ig1000	168	**90.1**	su1000	140	88.6	su100	172	80.5	su500	33	83.4
ig10000	113	89.1	su1000	167	**90.1**	ig1000	134	88.5	gr100	171	81.2	ig500	33	83.3
gr10000	112	89.1	gr1000	161	90.0	su500	117	88.2	ig100	166	79.8	ig1000	31	83.3
ig3000	93	88.9	gr3000	129	89.6	su3000	116	88.2	chi100	153	79.1	su1000	31	83.3
FS	WL	pr.	FS	WL	pr.	FS	WL	pr.	FS	WL	pr.	FS	WL	pr.
ig7000	158	88.9	ig1000	122	92.2	gr500	203	**93.6**	ig7000	146	**94.2**	gr500	50	87.7
su7000	158	88.9	su1000	120	92.2	gr1000	179	91.1	su7000	146	**94.2**	gr1000	48	89.0
chi7000	158	88.9	gr1000	120	92.2	gr100	130	89.9	gr7000	146	**94.2**	gr100	47	87.8
gr7000	157	88.9	chi1000	118	92.2	chi1000	70	89.4	chi7000	146	**94.2**	su100	7	87.1
ig5000	148	88.7	gr500	90	90.4	ig1000	68	89.4	gr100	93	89.4	ig100	2	86.9
FS	WL	re.	FS	WL	re.	FS	WL	re.	FS	WL	re.	FS	WL	re.
gr100	209	**98.2**	all	167	88.0	su1000	98	87.4	gr500	191	80.9	su500	33	78.8
gr500	187	95.9	chi1000	133	87.3	chi1000	97	87.4	su100	155	74.4	ig1000	32	78.8
all	134	92.7	ig1000	131	87.3	ig1000	92	87.3	ig100	153	74.1	su1000	32	78.8
gr1000	117	92.6	gr1000	131	87.2	su3000	86	87.0	chi100	138	73.6	chi500	32	78.8
su1000	117	92.6	su1000	130	87.3	ig3000	86	86.9	su500	126	72.9	ig500	31	78.8
FS	WL	F_1	FS	WL	F_1	FS	WL	F_1	FS	WL	F_1	FS	WL	F_1
all	162	**89.6**	chi1000	169	**89.7**	su1000	145	88.4	gr500	204	83.6	su500	38	82.4
chi10000	121	89.2	gr1000	166	**89.6**	chi1000	144	88.4	su100	175	79.1	ig500	37	82.4
gr10000	121	89.2	ig1000	164	**89.7**	ig1000	135	88.3	ig100	168	78.7	chi500	35	82.4
ig10000	120	89.2	su1000	163	**89.7**	su500	117	88.0	gr100	163	79.0	su1000	35	82.3
su10000	120	89.2	gr3000	129	89.2	su3000	114	88.0	chi100	156	78.3	ig1000	34	82.3
FS	WL	F_2	FS	WL	F_2	FS	WL	F_2	FS	WL	F_2	FS	WL	F_2
all	171	**91.4**	chi1000	153	88.2	chi1000	138	87.8	gr500	194	81.9	su500	35	80.2
gr1000	157	91.2	all	147	88.2	su1000	122	87.8	su100	162	76.2	ig500	34	80.2
su1000	148	91.1	ig1000	143	88.2	ig1000	120	87.7	ig100	154	75.9	chi500	34	80.2
ig1000	148	91.1	su1000	143	88.2	su500	95	87.3	chi100	140	75.4	ig1000	33	80.2
chi1000	148	91.1	gr1000	141	88.2	su3000	93	87.4	su500	131	74.6	su1000	33	80.2

Table 4. Top five feature selection methods, by accuracy, precision, recall, F_1 and F_2 wins–losses (WL), for each classifier with its "best" document representation

It can be seen from the table that different classifiers are best performers when different measures are considered. CNB and SMO are best at accuracy and the F_1 measure, VP and IBk are ahead of the field in precision, and CNB alone is the clear winner at F_2, and especially recall, with the staggering 98.2 % for GR at 100 selected features.

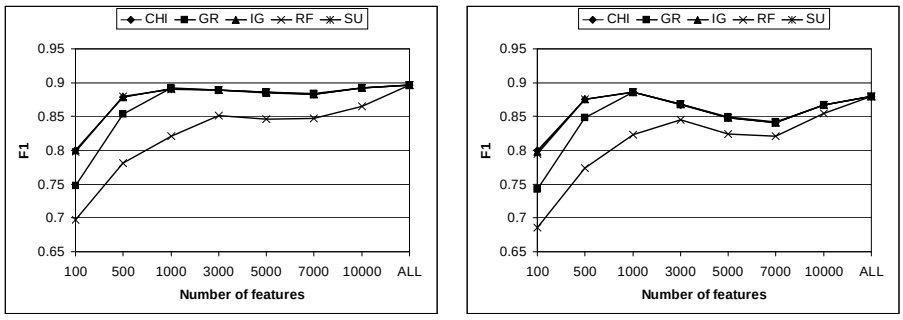
Considering feature selection methods, the tables indicate that CHI, IG and SU tend to "stick together," with similar performance at same reduction rates, while GR sometimes "breaks away" from the pack, although it is based on the same theoretical grounds as IG and SU. RF proved to be a complete failure, in all probability not because of any shortcoming as a feature selection algorithm, or unsuitability for use in textual domains. It was a consequence of WEKA's implementation using the Euclidean measure when calculating document vector distances for determining nearest neighbors.

In summary, considering its good all-round performance, dominance at recall and F_2, the lack of need for feature selection, and blazing speed, we can safely conclude that the CNB classifier, out of those considered, is best suited for the task of classifying search results. SMO can be regarded a close second, because of its inferior performance at F_2, and longer training times.

5.2.2 Interaction Between BOW Transformations and Feature Selection

This section will investigate how does the addition of normalization, *logtf* and *idf* transformations to a baseline BOW document representation affect classification performance, from the viewpoint of several feature selection methods. We will concentrate on two best-performing classifiers from Section 4: CNB and SMO. The baseline representation for normalization will be *m-tf* since stemming was beneficial to classification in all our previous experiments, while the baseline for *logtf* and *idf* will be the *m-norm-tf* representation because normalization is included in the determined best representations for the two classifiers. Since *idf* provided the motivation for this investigation, transformations will be presented in reverse order compared to Section 4: *idf*, *logtf*, and then *norm*.

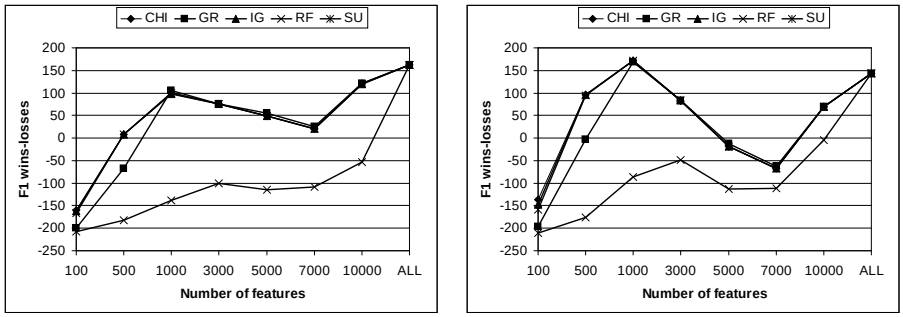
The *idf* transformation. Standard-style charts which show the F_1 measure of performance of CNB for feature selection algorithms and reduction rates are given in Figure 4. The measurements are averaged over the six datasets, and the used representations are *m-norm-tf* in Figure 4 a), and *m-norm-tfidf* in Figure 4 b). It can be seen, more clearly than in Section 5.2.1, that CHI, IG and SU feature selection methods exhibit almost identical behavior. GR follows the three down to the smaller numbers of features (500 and 100), where its performance decays. Since the described trends in the performance of FS methods were consistent over all evaluation measures with both classifiers, the following comments will generally refer to the CHI–IG–SU trio, unless stated otherwise.



a) b)

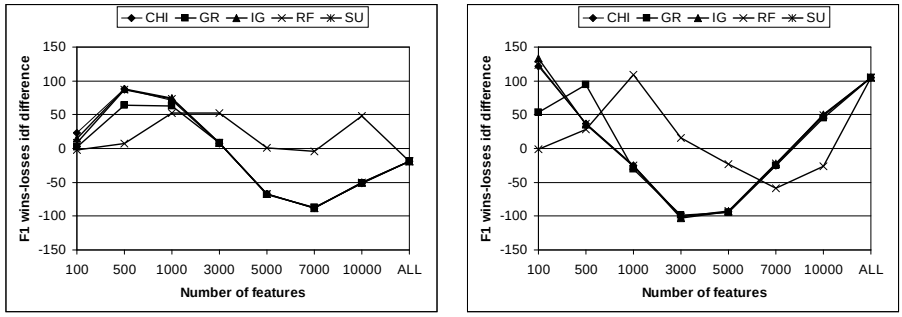
Fig. 4. Performance of CNB measured by F_1, with a) *tf* and b) *tfidf* representation

Besides CNB performance with *m-norm-tfidf* being several percent worse overall than with *m-norm-tf*, the only other noticeable difference between the two charts is the more pronounced dent between 3 000 and 10 000 features for *m-norm-tfidf*. However, when looking at the less used type of chart depicting summed-up statistically significant wins–losses in Figure 5, the differences become more obvious. These charts express how feature selection methods and reduction rates compare to one another in the context of the particular classifier (CNB) and representation, achieving independence from absolute measurements of performance. The peaks and dents are more clearly distinguishable than in Figure 4.



a) b)

Fig. 5. Summed up wins–losses for F_1 and CNB, with a) *tf* and b) *tfidf* representation



a) b)

Fig. 6. Effect of *idf* on F_1 wins–losses for a) CNB and b) SMO

Subtracting the wins–losses of the baseline *m-norm-tf* representation from the wins–losses of *m-norm-tfidf*, we obtained the chart in Figure 6 a). Effectively, it expresses the impact of the *idf* transformation on the performance of the CNB classifier (as measured by F_1) throughout the range of feature selection methods and reduction rates. According to the chart, *idf* improves CNB between 100 and 3 000 chosen features, while degrading it for higher feature counts. It should be

made clear that the improvement or degradation is only relative in nature, since using wins–losses limited the comparisons to the boundaries of the same document representation and classifier. Using wins–losses instead of actual performance measurements permitted the subtraction of values by avoiding the issue of scale when comparing the measurements on different document representations. Information about absolute performance was sacrificed in order to express the relationship between the *idf* transformation and feature selection. Therefore, the 100–3 000 range in Figure 6 a) can only be viewed as the place to *expect* improvement when introducing the *idf* transformation to the document representation. Whether there will be actual improvement is determined by the properties of classifiers and data. Our experience showed that *tfidf* representations are usually inferior for text categorization, which is certainly not a general rule [12, 17].

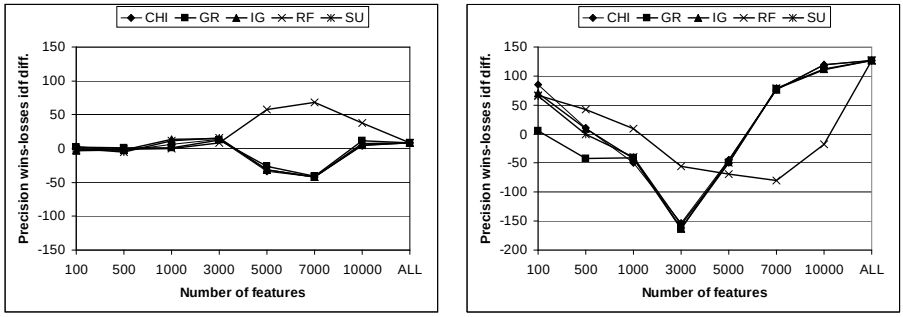
The corresponding chart of wins–losses differences for *idf* with SMO is shown in Figure 6 b). Two ranges with possible improvement of performance can be discerned: one approximately below 800 features, and another above 8 000. This demonstrates that the *idf* transformation has a different effect the two classifiers, and provides an explanation for the observed discrepancy. With no feature selection *idf* degrades CNB and improves SMO, while at 2 000–3 000 selected features the effect is opposite. (The 2 000–3 000 range roughly corresponds to 1 000 features from the previous batch of experiments since those datasets had a lower number of features.) What makes the analogy with the study from Section 4 even more remarkable is the fact that different datasets, and even feature selection methods were used.

The general shape of the graphs for CNB and SMO in Figure 6 (regarding the CHI–IG–SU trio) is quite similar, except for the drop of the CNB graph below 500 features. A corresponding drop for SMO may exist at a lower number of features which was not measured. This indicates that the CNB and SMO do not behave in such opposing fashion with regards to the *idf* transformation as was suggested, since the graphs are not totally contrary to one another.

However, observing the charts of wins–losses differences introduced by the *idf* transformation with CNB (Figure 7 a)) and SMO (Figure 7 b)) using *precision* as the evaluation measure reveals quite different effects of *idf* on the two classifiers.

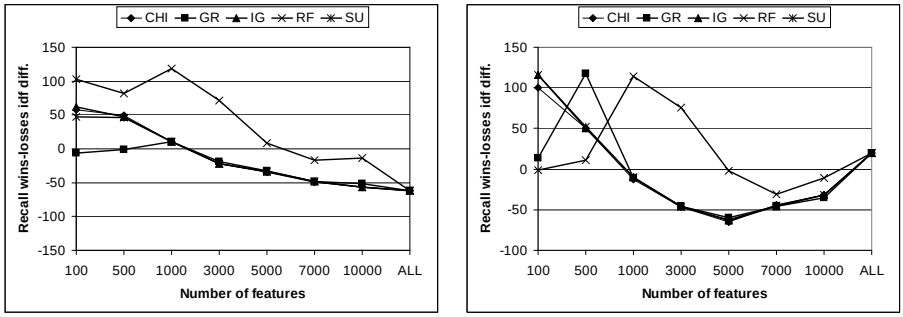
Since precision is one of the building blocks of the F_1 measure together with recall, this may suggest that the above correspondence may have been accidental, especially when additionally taking into consideration the charts for recall (Figure 8). But, we argue that the correspondence is not coincidental, since the wins–losses differences charts for *accuracy*, shown in Figure 9, are almost identical to those of F_1 (Figure 6).

The *logtf* transformation. The effects of the *logtf* transformation when measured by F_1, shown in Figure 10, are almost a complete contrast to those of *idf*. Generally, in the ranges where *idf* caused improvement of performance, *logtf* causes degradation, and vice versa. However, judging by the scale, *logtf* has a much milder



a) b)

Fig. 7. Effect of *idf* on *precision* wins–losses for a) CNB and b) SMO



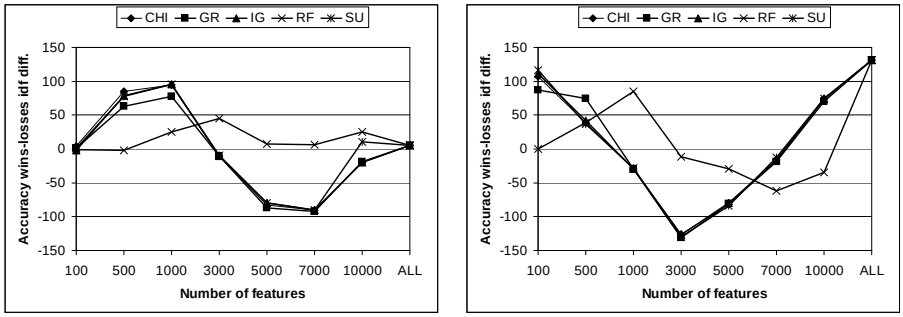
a) b)

Fig. 8. Effect of *idf* on *recall* wins–losses for a) CNB and b) SMO

impact than *idf*, especially on the CNB classifier. Again as with *idf*, the graphs for the two classifiers exhibit a certain degree of similarity.

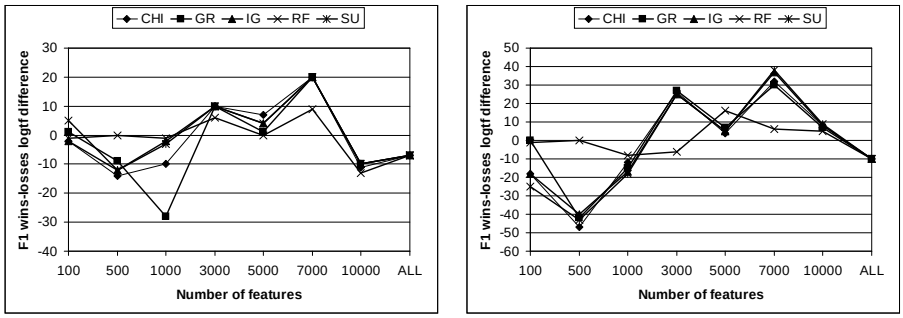
Looking at the charts showing the wins–losses differences with the precision measure it can be seen that *logtf* has a more clearly pronounced degrading effect on SMO in the feature range below 1 000, which accounts for the sharper drop in the same area on the F_1 chart. The scale in Figure 11 indicates that the effect of *logtf* on CNB is quite weak.

When observing recall in Figure 12, no notable differences between the effects of *logtf* on CNB and SMO are visible, except the generally weaker impact on CNB. Unlike *idf*, the *logtf* transformation provided no surprises and differences between its effects on different classifiers. The stronger impact on SMO, similar to the one already observed in Section 4, can be interpreted by the logarithm's smoothing effect on feature weights, and the scaling down to small numeric values. We find the *logtf* transformation more predictable and safer to use in the text categorization setting.



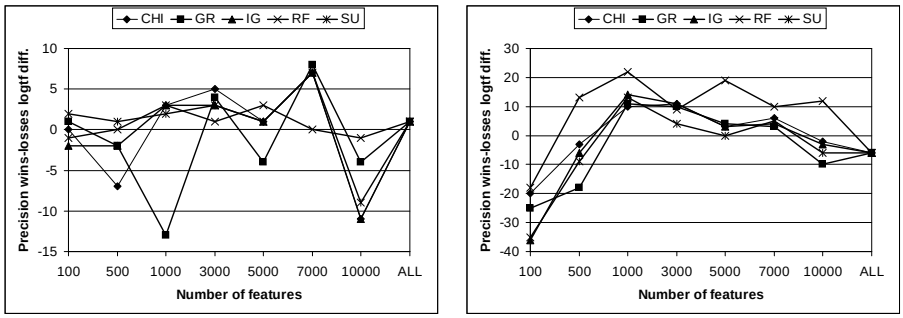
a) b)

Fig. 9. Effect of *idf* on *accuracy* wins–losses for a) CNB and b) SMO



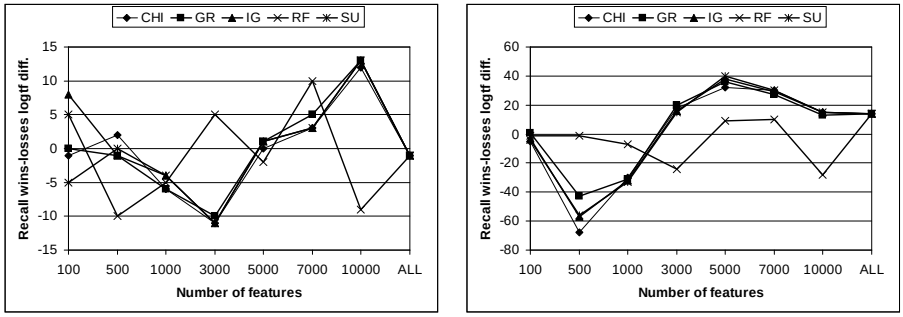
a) b)

Fig. 10. Effect of *logtf* on F_1 wins–losses for a) CNB and b) SMO



a) b)

Fig. 11. Effect of *logtf* on *precision* wins–losses for a) CNB and b) SMO



a) b)

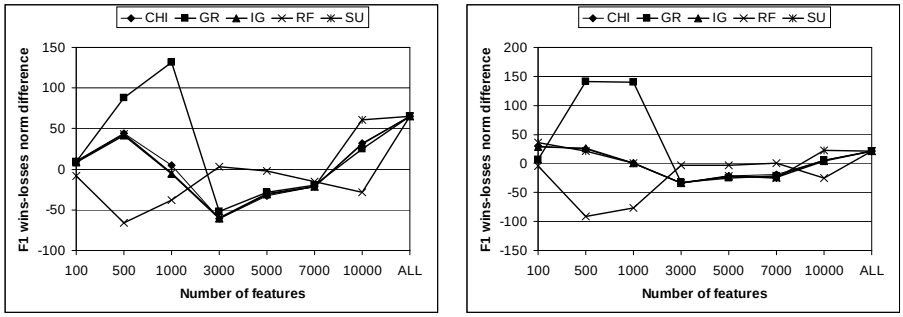
Fig. 12. Effect of *logtf* on *recall* wins–losses for a) CNB and b) SMO

The *norm* transformation. Figure 13 depicts the impact of normalization in the context of CNB and SMO classifiers. Somewhat counterintuitively, the shapes of the graphs are more similar to those of *idf* (Figure 6) than *logtf* (Figure 10). Compared to *idf*, the effects of normalization are much milder. Also, the behavior of the graphs for CNB and SMO classifiers in Figure 13 is quite similar. What is especially notable in the charts is the radical departure of the GR feature selection method from the behavior of CHI, IG and SU on feature counts below 3 000, in a similar fashion for both classifiers. However, the charts in Figures 14 and 15 reveal that the improving effect of GR measured by F_1 with the CNB classifier is caused by improvement of precision, while with SMO the main cause is recall. Overall, the gain ratio feature selection exhibited erratic behavior at lower feature counts with all investigated transformations.

6 CATS: A CLASSIFICATION-POWERED META-SEARCH ENGINE

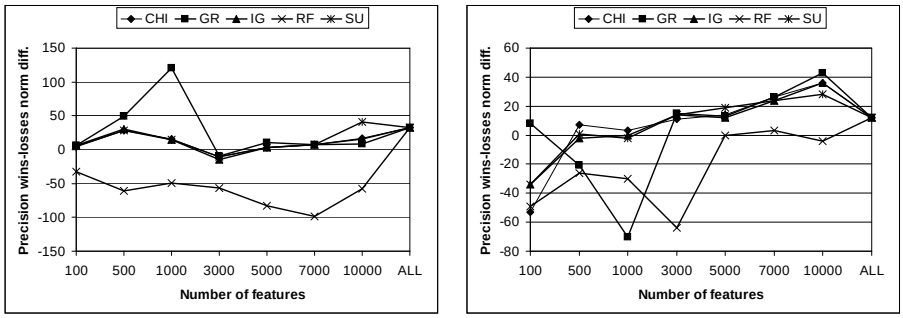
Traditionally, meta-search engines had been conceived in order to address different issues concerning general-purpose search engines. The issues include coverage of the Web, search result relevance, and their presentation to the user. A common approach to alternative presentation of results is by sorting them into (a hierarchy of) clusters which may then be displayed to the user in a variety of ways, e.g. as a separate expandable tree (`Vivisimo.com`) or arcs which connect Web pages within graphically rendered "maps" (`KartOO.com`). This section will discuss a new meta-search engine CatS (`stribog.pmf.uns.ac.rs/cats/`) the functionality of which is centered around sorting search results into a hierarchy of topics using *text categorization* instead of the more commonly employed clustering techniques.

The primary motivation for building CatS was to help the user resolve *query ambiguity* in a quick and convenient fashion. By query ambiguity we refer to the possibility of mismatch between the user's actual information need and the interpretation of query keywords by a search engine. One cause for ambiguity may be



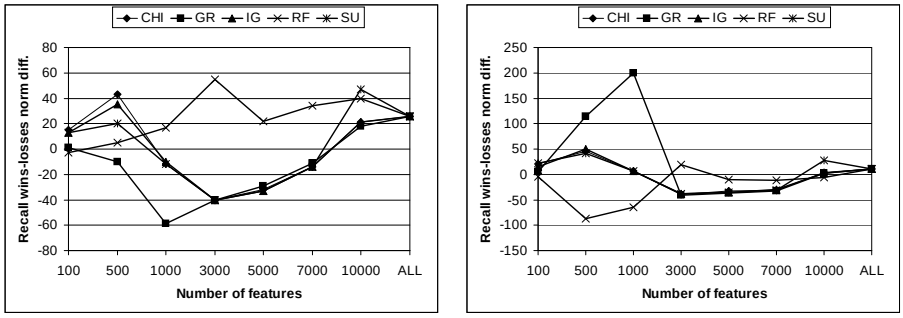
a) b)

Fig. 13. Effect of *norm* on F_1 wins–losses for a) CNB and b) SMO



a) b)

Fig. 14. Effect of *norm* on *precision* wins–losses for a) CNB and b) SMO



a) b)

Fig. 15. Effect of *norm* on *recall* wins–losses for a) CNB and b) SMO

different meanings of query keywords, in which case an unintended but more common meaning of a keyword may dominate the process of retrieval. Other causes of ambiguity may not have such intuitive explanations, and are in effect a result of the algorithms employed by search engines and their interaction with the content and structure of the Web. For example, the use of *inverse document frequencies* may cause a search engine to rank a document containing a rare keyword too high, as can "fancy hits," e.g. keywords appearing in Web page titles or as parts of e-mail addresses [10]. Most modern search engines rely on link-based algorithms like PageRank to determine authority and significance of Web pages, which can cause an irrelevant but extensively cited page to be ranked high in the search result list.

Nowadays, clustering is the dominant technique used by meta-search engines for alternative presentation of search results. However, the generated topic lists/hierarchies may not prove satisfactory for every query, and the "silver bullet" method has not yet been found. On the other hand, classification was considered for this task more than once ([19, p. 100], [28]), but was seldom actually tried.

6.1 The CatS System

CatS forwards a user's query to a major search engine, and augments the results with a browseable tree of topics. Figure 16 shows the subset of the 100 results for query 'animals england' sorted into category *Arts* → *Music*. Every category from the root (*All*) to the leaves can be clicked on, with the effect that the results belonging to the category are shown in the right frame of the displayed page.

The language chosen for the implementation of the system is Java. CatS executes as a servlet on an Apache Tomcat server which runs on a commodity PC under the FreeBSD operating system. The outline of CatS functionality is rather straightforward: A simple HTML form lets the user post a query and choose the major search engine to be used. The servlet forwards the query to the search engine in unmodified form and retrieves the HTML page of results. It then extracts the titles, hyperlinks and excerpts of all results, classifies the results based on their titles and excerpts, and shows the results together with the category tree to the user.

The categories employed by CatS were extracted and engineered from the *dmoz* Open Directory. Taking into account feasibility of implementation as well as practical usability of the system, two levels of categories were considered. From the top level a total of 11 categories were selected. The second level proved to be more of a challenge since some first-level topics contained, in our view, too many subtopics to be useful for search result classification. We therefore resorted to merging topics which appeared to be very related (e.g. *Artificial Intelligence* and *Robotics*) or overly specific (e.g. *Volleyball*, *Handball*, etc. were all merged into *Ball Sports*). Furthermore, some topics which contained a small number of examples were either merged with a related topic or discarded altogether. Table 5 summarizes all topics currently employed by CatS – the 11 top-level, and 146 second-level ones.

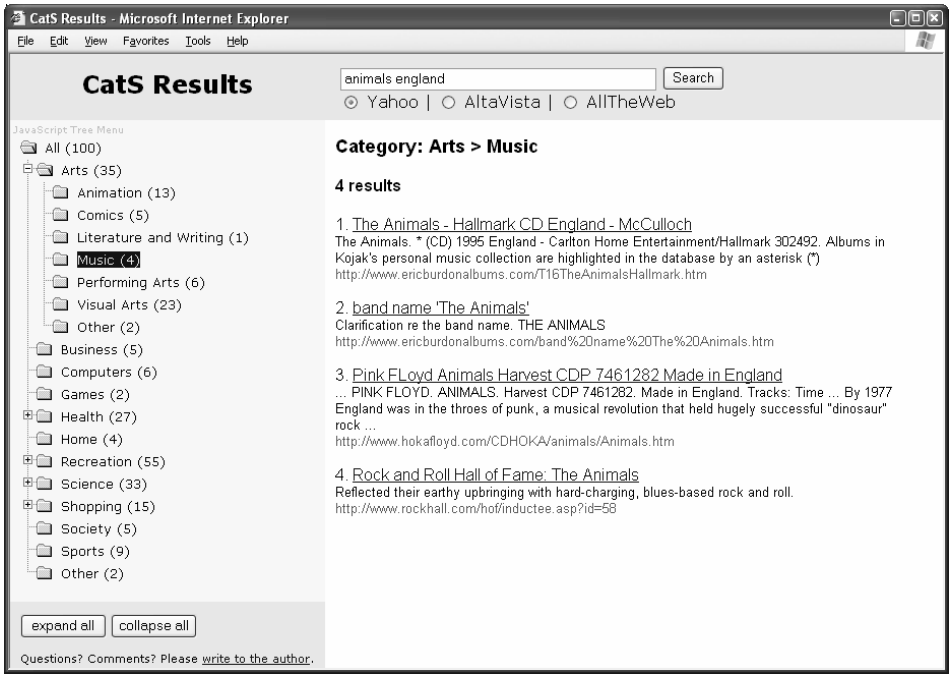


Fig. 16. Results for query 'animals england' classified into *Arts* → *Music*

Web-page titles and descriptions from *dmoz* were used to train binary classifiers for each chosen category. Experiences with the experiments described in Sections 4 and 5 provided strong guidelines for the choice of document representation, feature selection method and classification algorithm. Namely, we chose the normalized term frequency representation with stemming (*m-norm-tf*), no feature selection, and the ComplementNaiveBayes classifier.

At classification time, the binary classifier for each top-level category is executed on every search result. If more than n results populate a category (currently, n is fixed at 10), classification is continued at the second level for that top-level category. If a result does not classify as positive for any category (first or second level), it is sorted into category *Other* which is implicit at both levels. For more details about CatS, see [25].

7 CONCLUSION

The research that was presented in this paper can be divided into two strands: the experiments in text categorization (Sections 4 and 5) and the implementation of the meta-search engine CatS (Section 6).

Arts	Animation, Architecture, Art History, Bodyart, Comics, Crafts, Education, Literature and Writing, Movies, Music, Performing Arts, Radio, Television, Visual Arts
Business	Accounting and Financial Services, Agriculture and Environment, Arts and Entertainment, Automotive, Chemicals, Construction and Real Estate, Consumer Goods and Services, E-Commerce, Education and Training, Electronics and Electrical, Employment and Opportunities, Food and Related Products, Healthcare, Hospitality, Human Resources, IT and Telecommunications, Industrial Goods and Services, Management, Marketing and Advertising, Mining and Drilling, Publishing and Printing, Small Business, Textiles and Nonwovens, Trade, Transportation and Defence
Computers	Artificial Intelligence and Robotics, CAD and CAM, Computer Science, Data Formats, Education and E-Books, Graphics, Hardware, Internet, Multimedia, Open Source, Programming, Security and Hacking, Software, Systems
Games	Board Games, Card Games, Gambling, Miniatures, Online, Puzzles, Roleplaying, Video Games
Health	Adult Health, Alternative, Animal, Child and Teen Health, Conditions and Diseases, Fitness, Medicine, Mental Health, Nursing, Pharmacy, Professional, Public Health and Safety, Reproductive Health, Senior Health and Aging
Home	Consumer Information, Cooking, Family, Gardening, Homemaking, Personal Finance
Recreation	Animals, Collecting, Food, Humor, Models, Outdoors, Radio, Travel, Vehicles
Science	Agriculture, Astronomy, Biology, Chemistry, Earth and Environment Sciences, Educational Resources, Instruments and Supplies, Math, Philosophy, Physics, Social Sciences, Technology
Shopping	Antiques and Collectibles, Children, Clothing, Consumer Electronics, Entertainment, Food, General Merchandise, Gifts, Health, Home and Garden, Jewelry, Music, Pets, Publications, Sports and Recreation, Toys and Games, Vehicles, Visual Arts
Society	Activism, Crime, Death, Disabled, Ethnicity, Folklore, History, Holidays, Law, Paranormal, Philanthropy, Politics and Government, Relationships, Religion and Spirituality, Sexuality, Subcultures
Sports	Ball Sports, Equestrian, Fighting Sports, Gymnastics, Racing Sports, Strength Sports, Track and Field, Water Sports, Winter Sports

Table 5. All topics used by CatS

Section 4 experimentally demonstrated that there may be statistically significant differences in classification performance of five major classifiers when using different transformations of the bag-of-words document representation. The section also gave a description of the effects of individual transformations on five commonly used performance evaluation measures, indicating that the effects on different measures can be quite opposite. This was achieved by using wins–losses instead of absolute performance measure values, permitting manipulation such as addition and subtraction which would not otherwise have been possible. Furthermore, it was demonstrated that the effects and relationships can be significantly altered by the simple dimensionality reduction method that was used.

Section 5 built on the conclusions from Section 4 by considering a wider array of feature selection methods and reduction rates, but using only the document representations that were found most suitable for each classifier. Also, attention was focused on *idf*, the transformation that exhibited greatest variation of behavior with regards to feature selection in the previous section, within the context of two

best performing classifiers. The intuition that there may be significant interactions between the *idf* transformation and feature selection has been verified, and this interaction was quantified using charts of wins–losses and their differences. Similar treatment was given to two other transformations, *logtf* and *norm*, also revealing interesting effects, but less radical and erratic than those of *idf*. Together, the two experimental sections helped determine the best combination of document representation, feature selection method and classifier for use in the CatS meta-search engine, presented in detail in Section 6.

Experience with CatS and other available meta-search engines has shown that using classification for sorting search results has both its advantages and disadvantages. Categories are chosen by humans, rather than constructed by some clustering algorithm, so it is clear what their meanings are, and the user will most probably find browsing comfortable and intuitive. But, this clarity makes classification errors obvious, possibly reducing the user's faith in the system. This effect was emphasized for CatS by the choice to prefer overpopulating categories over leaving results unclassified. Furthermore, a fixed set of topics may not suit every query: for example, CatS sorts almost all results for 'java servlets' under *Computers* → *Programming*, which is unlikely to be particularly useful if computer programming is precisely what the user is interested in. Clustering approaches to sorting search results tackle this issue by dynamically generating topics from the list of results for every individual query. But, while for some queries the identified topics look natural and helpful, for others they may seem somewhat ambiguous.

Further work on the issues raised in the paper can mostly be carried out in an independent fashion within the described research paths.

Within the experimental strand, identical experiments can be performed on more commonly used corpora, in order to draw parallels with existing research and make sure that the conclusions are not relevant only to our datasets. Giving more transformations besides *idf*, *logtf* and *norm* the treatment of Section 5.2.2 may also reveal interesting relationships, especially for transformations based on supervised learning [4]. Finally, the proposed methods can be used in the context of more advanced document representation schemes, which may include n-grams (as sequences of n words), hyperlink information, and elements of document structure.

From the point of view of CatS, the possibilities of further work are numerous: systematic engineering of categories, adding more topic levels, aggregating results from several search engines, trying out different user interfaces, and, most notably, investigating a mixture of classification and clustering techniques for sorting results in order to achieve a marriage of the clarity and familiarity of human-engineered topics with the dynamic nature and adaptability of automatically generated clusters. In an ideal scenario, classification technology could be integrated into Web-page collections like the ones maintained by general-purpose search engines. Then, classification performance would be enhanced not only by the availability of full Web-page texts, but also by consulting link data used to calculate page rankings. Furthermore, in the spirit of vertical search, results could be re-ranked depending on the category being viewed.

To summarize, the work presented in this paper gave a particular angle on the application of machine learning techniques on the Web, initially motivated by the intention of building the CatS meta-search engine. Experiments were carried out in order to determine the needed parameters for the implementation of CatS. But, both the experiments and CatS outgrew their initial rationales, reaching conclusions and raising issues that were not initially anticipated.

Acknowledgments

The authors would like to thank the handling editor and anonymous reviewers for helping to greatly improve the paper. The work was supported by the project "Abstract Methods and Applications in Computer Science" (No. 144017A), of the Serbian Ministry of Science and Technological Development.

REFERENCES

[1] Aha, D.—Kibler, D.—Albert, M. K.: Instance-Based Learning Algorithms. Machine Learning, Vol. 6, 1991, No. 1, pp. 37–66.

[2] Chen, H.—Dumais, S. T.: Bringing Order to the Web: Automatically Categorizing Search Results. In: Proc. Human Factors in Computing Systems, CHI '00, 2000, pp. 145–152.

[3] Carpineto, C.—Romano, G.: Exploiting the Potential of Concept Lattices for Information Retrieval with CREDO. Journal of Universal Computer Science, Vol. 10, 2004, No. 8, pp. 985–1013. CREDO is accessible at `credo.fub.it/`.

[4] Debole, F.—Sebastiani, F.: Supervised Term Weighting for Automated Text Categorization. In: S. Sirmakessis (Ed.): Text Mining and its Applications, Studies in Fuzziness and Soft Computing, Vol. 138, Physica-Verlag, Heidelberg, Germany, 2004, pp. 81–98.

[5] Ferragina, P.—Gulli, A.: A Personalized Search Engine Based on Web-Snippet Hierarchical Clustering. In: Proc. 14th International World Wide Web Conference, WWW '05, Chiba, Japan, 2005, pp. 801–810.
SnakeT is accessible at `snaket.di.unipi.it/`.

[6] Forman, G.: An Extensive Empirical Study of Feature Selection Metrics for Text Classification. Journal of Machine Learning Research, Vol. 3, 2003, pp. 1289–1305.

[7] Freund, Y.—Schapire, R. E.: Large Margin Classification Using the Perceptron Algorithm. Machine Learning, Vol. 37, 1999, No. 3, pp. 277–296.

[8] Gabrilovich, E.—Markovitch, S.: Text Categorization with Many Redundant Features: Using Aggressive Feature Selection to Make SVMs Competitive with C4.5. In: Proc. 21st International Conference on Machine Learning, ICML '04, Banff, Canada, 2004, pp. 41–48.

[9] Hall, M. A.—Smith, L. A.: Practical Feature Subset Selection for Machine Learning. In: Proc. 21st Australasian Computer Science Conference, Perth, Australia, 1998, pp. 181–191.

[10] Jackson, P.—Moulinier, I.: Natural Language Processing for Online Applications: Text Retrieval, Extraction and Categorization. John Benjamins, 2002.

[11] Joachims, T.: Making Large-Scale SVM Learning Practical. In: B. Scholkopf, C. Burges and A. Smola (Eds.): Advances in Kernel Methods – Support Vector Learning, MIT Press, 1999, pp. 169–184.

[12] Kibriya, A. M.—Frank, E.—Pfahringer, B.—Holmes, G.: Multinomial Naive Bayes for Text Categorization Revisited. In: Proc. 17th Australian Joint Conference on Artificial Intelligence, AI '04, Cairns, Australia, 2004, Lecture Notes in Artificial Intelligence, Vol. 3339, pp. 488–499.

[13] Kira, K.—Rendell, L.: A Practical Approach to Feature Selection. In: Proc. 9th International Workshop on Machine Learning, Aberdeen, Scotland, 1992, pp. 249–256.

[14] Koester, B.: Conceptual Knowledge Retrieval with FooCA: Improving Web Search Engine Results with Contexts and Concept Hierarchies. In: Proc. 6th Industrial Conference on Data Mining, ICDM '06, Leipzig, Germany, 2006, Lecture Notes in Artificial Intelligence, Vol. 4065, pp. 176–190. FooCA is accessible at `fooca.webstrategy.de/`.

[15] Kononenko, I.: Estimating Attributes: Analysis and Extensions of RELIEF. In: Proc. European Conference on Machine Learning, ECML94, Catania, Italy, 1994, Lecture Notes in Computer Science, Vol. 784, pp. 171–182.

[16] Lawrie, D.—Croft, W. B.: Generating Hierarchical Summaries for Web Searches. In: Proc. 26th ACM International Conference on Research and Development in Information Retrieval, SIGIR '03, Toronto, Canada, 2003, pp. 457–458. CiiRarchies is accessible at `www.cs.loyola.edu/~lawrie/hierarchies/`.

[17] Leopold, E.—Kindermann, J.: Text Categorization with Support Vector Machines. How to Represent Texts in Input Space? Machine Learning, Vol. 46, 2002, pp. 423–444.

[18] Mitchell, T. M.: Machine Learning. McGraw-Hill, 1997.

[19] Mladenić, D.: Machine Learning on Non-Homogenous, Distributed Text Data. Ph. D. thesis, University of Ljubljana, Slovenia, 1998.

[20] Mladenić, D.: Text-Learning and Related Intelligent Agents. IEEE Intelligent Systems, Vol. 14, 1999, No. 4, pp. 44–54.

[21] Osiński, S.—Weiss, D.: A Concept-Driven Algorithm for Clustering Search Results. IEEE Intelligent Systems, Vol. 20, 2005, No. 3, pp. 48–54. Carrot2 is accessible at `www.carrot2.org/`.

[22] Osuna, E.—Freund, R.—Girosi, F.: An Improved Training Algorithm for Support Vector Machines. In: Proc. 7th IEEE Neural Networks for Signal Processing Workshop, Pisctaway, NJ, 1997, pp. 276–285.

[23] Platt, J.: Fast Training of Support Vector Machines Using Sequential Minimal Optimization. In: B. Scholkopf, C. Burges and A. Smola (Eds.): Advances in Kernel Methods – Support Vector Learning, MIT Press, 1999, pp. 185–208.

[24] Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.

[25] Radovanović, M.—Ivanović, M.: CatS: A Classification-Powered Meta-Search Engine. In: M. Last et al. (Eds.): Advances in Web Intelligence and Data Mining,

Studies in Computational Intelligence, Vol. 23, Springer-Verlag, 2006, pp. 191–200. CatS is accessible at stribog.pmf.uns.ac.rs/cats/.

[26] Radovanović, M.—Ivanović, M.: Document Representations for Classification of Short Web-page Descriptions. In: Proc. 8th International Conference on Data Warehousing and Knowledge Discovery, DaWaK'06, Krakow, Poland, 2006, Lecture Notes in Computer Science, Vol. 4081, pp. 544–553.

[27] Rennie, J. D. M.—Shih, L.—Teevan, J.—Karger, D. R.: Tackling the Poor Assumptions of Naive Bayes Text Classifiers. In: Proc. 20th International Conference on Machine Learning, ICML'03, 2003, pp. 616–623.

[28] Sebastiani, F.: Machine Learning in Automated Text Categorization. ACM Computing Surveys, Vol. 34, 2002, No. 1, pp. 1–47.

[29] Sebastiani, F.: Text Categorization. In: A. Zanasi (Ed.): Text Mining and its Applications, WIT Press, Southampton, UK, 2005, pp. 109–129.

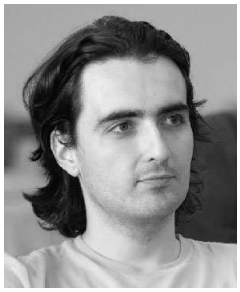
[30] Vapnik, V.—Chervonenkis, A.: Theory of Pattern Recognition (in Russian). Nauka, 1974.

[31] Witten, I. H.—Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Second Edition, Morgan Kaufmann, 2005.

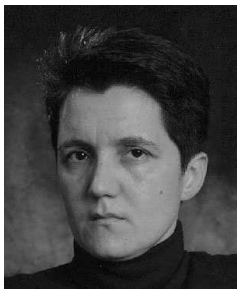
[32] Wu, X.—Srihari, R.—Zheng, Z.: Document Representation for One-Class SVM. In: Proc. 15th European Conference on Machine Learning, ECML'04, Pisa, Italy, 2004, Lecture Notes in Artificial Intelligence, Vol. 3201, pp. 489–500.

[33] Wu Y.-F.—Chen, X.: Extracting Features from Web Search Returned Hits for Hierarchical Classification. In: Proc. International Conference on Information and Knowledge Engineering, IKE'03, Las Vegas, Nevada, USA, 2003, pp. 103–108. Highlight is accessible at highlight.njit.edu/.

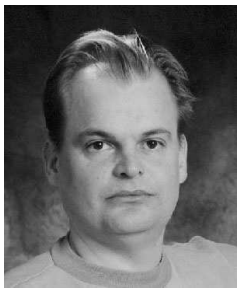
[34] Yang, Y.—Pedersen, J. O.: A Comparative Study on Feature Selection in Text Categorization. In: Proc. 14th International Conference on Machine Learning, ICML'97, Nashville, US, 1997, pp. 412–420.



Miloš Radovanović is a Teaching Assistant at the Faculty of Science, Department of Mathematics and Informatics, University of Novi Sad, where he graduated in 2001 and received his master's degree in informatics in 2006. Currently, the fields of his research interest are data mining, web mining, and text categorization. He published more than 20 research papers in refereed proceedings and journals, one university textbook and two textbook chapters. He is presently pursuing a doctor's degree at the Department of Mathematics and Informatics.



Mirjana Ivanović is a Professor at the Faculty of Science, Department of Mathematics and Informatics, University of Novi Sad. She graduated in 1983 (informatics), received master's degree (discrete mathematics and programming) in 1988 and doctor's degree (computer science) in 1992. Her research interests include: multi-agent systems, e-learning and web-based learning, data mining, case-based reasoning, programming languages and tools. She actively participates in more than 10 international and several national projects. She published over 120 scientific papers in proceedings of international conferences and journals, and wrote more than 10 university textbooks in the field of informatics and information and communication technologies. She is the head of the computer science chair at the Department of Mathematics and Informatics.



Zoran Budimac is a Professor at the Faculty of Science, Department of Mathematics and Informatics, University of Novi Sad. He graduated in 1983 (informatics), received master's degree (computer science) in 1991 and doctor's degree (computer science) in 1994. His research interests include: mobile agents, e-learning, software engineering, case-based reasoning, implementation of programming languages. He has been project leader for several international and several national projects. He published over 130 scientific papers in proceedings of international conferences and journals, and wrote more than 10 university textbooks in different fields of informatics.