A hierarchical text categorization approach and its application to FRT expansion [1]

Domonkos Tikk [a,b,2] György Biró [c,a] Jae Dong Yang [d]

[a] *Department of Telecommunications & Telematics, Budapest University of Technology and Economics, H-1117 Budapest, Magyar Tudósok Körútja 2., Hungary*

[b] *Integrated Intelligent Systems Japanese–Hungarian Laboratory H-1111 Budapest, Műegyetem rakpart 3., Hungary*

[c] *Department of Informatics, Eötvös Loránd Science University, H-1117 Budapest, Pázmány sétány 1/c, Hungary*

[d] *Dept. of Computer Science, Chonbuk National University Chonju 561–756, Korea*

Abstract

Text categorization is the classification to assign a text document to an appropriate category in a predefined set of categories. This paper focuses on the special case when categories are organized in hierarchy. We presents a new approach on this recently emerged subfield of text categorization. The algorithm applies an iterative learning module that allow of gradually creating a classifier by trial-and-error-like method. Experimental results performed on three document corpora (including the well-known Reuters-21578, and 20 newsgroups data sets) with several topic hierarchies show that our approach outperforms existing ones by up to 10%. We also indicate another application of the method on the field of fuzzy relational thesauri (FRT): the expansion of knowledge base can be supported in a cost-effective way.

Key words: Text mining; Multi-level categorization; Hierarchical text categorization; Fuzzy relational thesaurus; Knowledge base expansion.

Email addresses: `tikk@ttt.bme.hu` (Domonkos Tikk), `gbiro@ttt.bme.hu` (György Biró), `jdyang@cs.chonbuk.ac.kr` (Jae Dong Yang).

[1] Research funded by the Hungarian Scientific Research Fund Grant No. D034614 and by the Hungarian Ministry of Education Grant No. FKFP 0180/2001. This research was partly sponsored by the Chonbuk National University, Korea

[2] Corresponding author

1 Introduction

With the advent of data warehouses, the importance of text mining has been ever increasing in the last decade. A significant subfield of text mining is text document categorization that aims at the automatic classification of electronic documents. Text categorization is the classification to assign a document to appropriate category/ies, also called *topic*, in a predefined set of categories.

Traditionally, document categorization has been performed manually. However, as the number of documents explosively increases, the task becomes no longer amenable to the manual categorization, requiring a vast amount of time and cost. This has lead to numerous researches for automatic document classification.

Originally, research in text categorization addressed the *binary problem*, where a document is either relevant or not w.r.t. a given category. In real-world situation, however, the great variety of different sources and hence categories usually poses *multi-class* classification problem, where a document belongs to exactly one category selected from a predefined set [3,11,16,29,31,32]. Even more general is the case of *multi-label* problem, where a document can be classified into more than one category. While binary and multi-class problems were investigated extensively [25], multi-label problems have received very little attention [1].

As the number of topics becomes larger, multi-class categorizers face the problem of complexity that may incur rapid increase of time and storage, and compromise the perspicuity of categorized subject domain. A common way to manage complexity is using a hierarchy [3], and text is no exception [4]. Internet directories and large on-line databases are often organized as hierarchies; see e.g. Yahoo and IBM's patent database [4]. Other real-world applications also often pose problems with hierarchical category classification, such as sorting of e-mails and/or files into folder hierarchies, structured search and/or browsing, etc.

Text categorization into topic hierarchies, also called *taxonomies*, is a particular type of multi-label classification problem. A document belonging to a topic in the taxonomy also belongs to all of its parent topics along a *topic path*. As a consequence, categories of a document can be subsequently determined at each level going downward in the taxonomy. This feature saves time considerably since at a time one has to select the best category only from a few one. Namely, once having selected a topic at a certain level in the hierarchy, only its

[3] In general hierarchy is considered to be an acyclic digraph; in this paper we restrict our investigation to tree structured hierarchies.
[4] `http://www.yahoo.com`, `http://www.ibm.com/patents`

children should be considered as prospective categories at the next level. Given a three level taxonomy and an average of 10 children at each node, the search method described reduces the number of considered categories from 1000 to 30. To exemplify the classification problem and terminology of our approach consider the subject domain of *professional basketball and baseball news*. The three level taxonomy of is depicted in Figure 2. Further, let us consider a document classified into the topic `Team News (Rams)`. The topic path of the document is `NFL` $\rightarrow$ `Rams` $\rightarrow$ `Team New (Rams)`. We aims at classifying the the document downward in the taxonomy to find the right category.

This paper describes a hierarchical text categorization approach. The main part of the approach is an iterative learning module that gradually trains the classifier to recognize constitutive characteristics of categories and hence to discriminate typical documents belonging to different categories. The iterative learning helps to avoid overfitting of training data and to refine characteristics of categories.

Characteristics of categories are captured by typical terms occurring frequently in documents assigned to them. We represent categories by weight vectors, called *category descriptors* (or simply descriptors), assigned to terms occurring in the document collections. The higher weight a term has in a category descriptor the more significant it is to the given category. Category descriptors are tuned during the iterative learning based on the their sample training documents.

We report on our experience with our approach on three document corpora: the Reuters-21578 newswire benchmark used widely in the information retrieval (IR) community, the 20-newsgroups data set, also having been studied by many authors (see e.g. [3,17,24]) and on the TV closed caption data set [6] (courtesy of W. Chuang). The effectiveness of our classifier is 67–94% depending on the corpus and the percentage of the training document used. These results are superior to the best known numbers in the literature.

We also present another possible application area of our approach. This is the knowledge base expansion of fuzzy relational thesauri (FRT). A thesaurus in an information retrieval system (IRS) can be considered as a *knowledge base* that represents the conceptual model of certain subject domain [14,18,21]. FRT are usually created manually that is a rather time inefficient and costly process. Exploiting that in fuzzy thesauri concepts are usually organized into a hierarchy, we can adapt our approach to support the creation of FRT using the concept hierarchy of FRT as a topic hierarchy. Given an initial FRT consisting only of a few concepts, we can expand it by adding the most typical terms of selected topics.

The paper is organized as follows. Section 2 reviews the related works of text categorization. Section 3 is the main part of this paper our approach is detailed. Section 4 report on our experience and Section 5 shows possible application to FRT expansion. The conclusion is drawn in Section 6.

2 Related works

Numerous statistical classification and machine learning techniques have been applied to text categorization. They include nearest neighbor classifiers (KNN) [32], regression models [32], voted classification [29], Bayesian classifiers [16], decision trees [16], Support Vector Machines (SVM) [11], information-theoretic approaches (e.g. distributional clustering) [3] and neural networks [31]. For comparative studies see [1,25]. Usually these techniques are compared on a standardized collection of documents, such as the Reuters newswire corpus. The results cannot be directly compared as different papers applied different versions of the document collection [25, page 38] and tested the performance by different measures. Nevertheless, we can state that some version of KNN, SVM and voted classification provide the best results, achieving around 86.3–87.8 percentage for precision-recall break-even points (results may somewhat vary at different authors). The overall greatest break-even point, 87.8, was attained by Weiss et al [29]. Their method uses decision trees induced by means of adaptive-resampling algorithm and pooled local dictionaries. To determine the category of a document, Weiss' approach applies voting to multiple decision trees.

On the other hand, hierarchical text categorization is a recently emerged topic of text mining. Before the result of Koller and Sahami in 1997 [12], there has been only some work on hierarchical clustering, e.g. [9]. In [12] the authors focused on the reduction of *feature set*, i.e. they aimed at minimizing the number of *terms* that were used to discriminate between categories. They used Bayesian classifier and allowed dependencies between features. Their results experimented on two small subsets of the Reuters collection shows that hierarchical classifiers outperform flat ones when the number of features is small (less than 100). Their approach was criticized in e.g. [17], because it did not show improvement with larger dictionaries, although in many domains it has been established that large dictionary sizes often perform best [11,17,20].

Hierarchical text categorization has been combined in many works with feature subset selection that improves classification accuracy, reduces measurement cost, storage and computational overhead by finding the best subset of features [6]. As examples, TAPER [4] employs a taxonomy and classifies text using statistical pattern recognition techniques. It finds feature subset by the Fisher's discriminant. In [19], under the simplified assumption of Koller and Sahami, authors used naïve Bayesian classifier combined with feature subset of *n-grams*. McCallum *et al.* also used the naïve classifier [17]. They adopted an established statistical technique called *shrinkage* to improve parameter estimates of class probabilities in taxonomy. A simple but fast solution was proposed in [6], where TFIDF classifier [13,22] (using tf$\times$idf weighting, see (3)) was applied for hierarchical classification. They applied a greedy algorithm at each level of the hierarchy that resulted in $O(n \log n)$ time for n documents.

All referred results on hierarchical classifier showed superior performance to flat ones. Straightforward comparison of these methods stumbles over the lack of a unique document collection. These methods have been tested on different and sometimes even provisional (e.g. web pages) text corpora. We report on experience with our approach on Reuters-21578 in comparison with the papers of D'Alessio et al [7] and Chakrabarti et al [4], on 20 newsgroups data set in comparison with the paper of McCallum et al [17], and Wibowo and Williams [30], and on TV closed caption data set in comparison with the paper of Chuang et al [6] (see Section 4).

3 The proposed method

The core idea of the categorization method is the training algorithm that sets and maintains category descriptors in a way that allows the classifier to be able to correctly categorize the most training, and consequently, test documents. We start the categorization with empty descriptors.

We now briefly describe the training procedure. First, when classifying a training document we compare it with category descriptors. When this procedure fails due to, e.g., the small size of descriptors, and the correct category of a document could not be determined we raise the weight of such terms in the category descriptors that appear also in the given document. Contrary, if a document is assigned to a category incorrectly, we lower the weight of such terms in the descriptors that appear in the document. We tune category descriptors by finding the optimal weights for each terms in each category descriptor by this awarding–penalizing method. The training algorithm is executed iteratively and ends when the performance of the classifier cannot be further improved significantly. See the block diagram of Figure 1 for an overview and details in Subsection 3.2 about the training algorithm. For test documents the classifier works in one pass by omitting the feedback cycle.

The rest of this section is organized as follows. Subsection 3.1 describes the topic hierarchy, vector space model and descriptors. Subsection 3.2 presents classification and the training method.

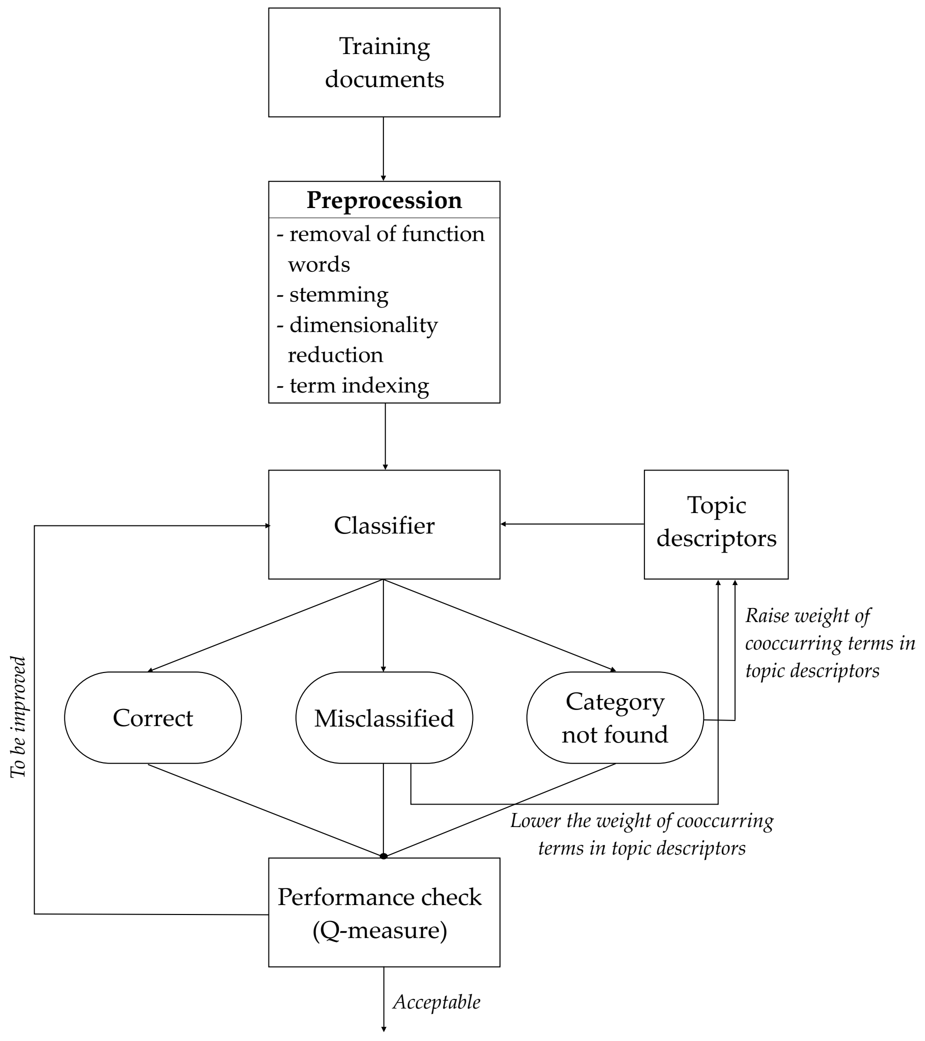
Training documents
Preprocession
- removal of function words
- stemming
- dimensionality reduction
- term indexing
Classifier
Topic descriptors
Raise weight of cooccurring terms in topic descriptors
To be improved
Correct
Misclassified
Category not found
Lower the weight of cooccurring terms in topic descriptors
Performance check (Q-measure)
Acceptable

Fig. 1. The flowchart of the training algorithm

3.1 Definitions

Let $\mathcal{C}$ be the fixed finite set of categories organized in a *topic hierarchy.* In this paper, we deal with tree structured topic hierarchies, and we do not allow multiple parentcraft unlike in our previous works [27,26].

Let $\mathcal{D}$ be a set of text documents and $d \in \mathcal{D}$ an arbitrary element of $\mathcal{D}$. In general, documents are pre-classified under the categories of $\mathcal{C}$, in our case into leaf categories. We differentiate training, $d \in \mathcal{D}_{\text{Train}}$, and test documents, $d \in \mathcal{D}_{\text{Test}}$, where $\mathcal{D}_{\text{Train}} \cap \mathcal{D}_{\text{Train}} = \emptyset$, and $\mathcal{D}_{\text{Train}} \cup \mathcal{D}_{\text{Train}} = \mathcal{D}$. Training documents are used to inductively construct the classifier. Test documents are used to test the performance of the classifier. Test documents do not participate in the construction of the classifier in any way.

Each document $d_j \in \mathcal{D}$ is classified into a leaf category of the hierarchy. No document belongs to non-leaf categories. We assume that a parent category owns the documents if its child categories, i.e., each document belongs to a *topic path* containing the nodes (representing categories) from the root to a leaf. Formally,

$$\text{topic}(d_j) = \{c_1, \ldots, c_q \in \mathcal{C}|\} \tag{1}$$

determines the set of topics d_j belongs to along the topic path from the highest to the deepest. Note that the root is not administrated in the topic set, as it owns all documents. c_q denotes leaf-category, and the index refers to the depth of the category.

Texts cannot be directly interpreted by a classifier. Because of this, an indexing procedure that maps a text d into a compact representation of its content needs to be uniformly applied to all documents (training and test). We choose to use only words as meaningful units of representing text, because, the use of n-grams (word sequences of length n) increases dramatically the storage requirement of the model, and as it was reported in [2,8] the use of more sophisticated representation than simple words do not increase effectiveness significantly.

As most research works, we also use the *vector space* model, where a document d_j is represented by a vector of term *weights*

$$d_j = (w_{1j}, \ldots, w_{|\mathcal{T}|j}), \tag{2}$$

where $\mathcal{T}$ is the set of *terms* that occurs at least ones in the training documents $\mathcal{D}_{\text{Train}}$, and $0 \leq w_{kj} \leq 1$ represents the relevance of kth term to the characterization of the document d. Before indexing the documents *function words* (i.e. articles, prepositions, conjunctions, etc.) are removed, and stemming (grouping words that share the same morphological root) is performed on $\mathcal{T}$. We utilize the well-known tf×idf weighting [23], which defines w_{kj} in proportion to the number of occurrence of the kth term in the document, o_{kj}, and in inverse proportion to the number of documents in the collection for which the terms occurs at least once, n_k:

$$w_{kj} = o_{kj} \cdot \log\left(\frac{N}{n_k}\right), \tag{3}$$

Term vectors (2) are normalized before training.

We characterize categories analogously as documents. To each category is assigned a vector of *descriptor term weights*

$$\text{descr}(c_i) = \langle v_{1i}, \ldots, v_{|\mathcal{T}|i} \rangle, \quad c_i \in \mathcal{C} \tag{4}$$

where weights $0 \leq v_{1i} \leq 1$ are set during training. All weights are initialized as 0. The descriptor of a category can be interpreted as the prototype of a document belonging to it.

3.2 *Classification and training*

3.2.1 *Classification*

When classifying a document $d \in \mathcal{D}$ the term vector representing d (2) is compared to topic descriptors (4). The vector of d is matched against a set of descriptors and based on the result the classifier selects (normally) a unique category.

The classification method works downward in the topic hierarchy level by level. First, it determines the best among the top level categories. Then its children categories are considered and the most likely one is selected. Considered categories are always siblings linked under the *winner category* of the previous level. Classification ends when a leaf category is found.

Let us assume that we have to select from k categories at an arbitrary stage of the classification of document d_j: $c_1, \ldots, c_k \in \mathcal{C}$. Then we calculate the conformity of term vector of d_j and each topic descriptors $\mathrm{descr}(c_1), \ldots, \mathrm{descr}(c_k)$, and select that category that gives the highest conformity measure. We applied the unnormalized cosine measure that calculates this value as a function f of the sum of products of document and descriptor term weights:

$$\mathrm{conf}(d_j, \mathrm{descr}(c_i)) = f\left(\sum_{k=1}^{|\mathcal{T}|} w_{kj} \cdot v_{ki}\right), \tag{5}$$

where $f : \mathbb{R} \to [0,1]$ is an arbitrary smoothing function with $\lim_{x\to 0} f(x) = 0$ and $\lim_{x\to\infty} f(x) = 1$. The smoothing function is applied (analogously as in control theory) to alleviate the oscillating behavior of training.

McCallum [17] criticized the greedy topic selection method because it requires high accuracy at internal (non-leaf) nodes. In order to alleviate partly the risk of a high level misclassification, we control the selection of the best category by a minimum conformity parameter $\mathrm{conf}_{\mathrm{min}} \in [0,1]$, i.e. the greedy selection algorithm continues when

$$\mathrm{conf}(d_j, \mathrm{descr}(c_{\mathrm{best}})) \geq \mathrm{conf}_{\mathrm{min}}$$

satisfied, where c_{best} is the best category at the given level.

3.2.2 *Training*

In order to improve the effectiveness of classification, we apply supervised iterative learning, i.e. we check the correctness of the selected categories for training documents and if necessary, we modify term weights in category descriptors. Term weights are modified when a document is classified incorrectly.

The classifier can commit two kinds of errors: it can misclassify a document d_j into c_i, and usually simultaneously, it cannot determine the correct category of d_j. Our weight modifier method is able to cope with both types of error. We scan all decision made by the classifier and process as follows.

For each considered category c_i at a given level we accumulate a vector $\delta(c_i) = \langle \delta(v_{1i}), \ldots, \delta(v_{\mathcal{T}i}) \rangle$ where

$$\delta(v_{ki}) = \alpha \cdot (\text{conf}_{\text{req}} - \text{conf}(d_j, \text{descr}(c_i)))) \cdot w_{ij}, \quad 1 \le k \le \mathcal{T} \tag{6}$$

where $\text{conf}_{\text{req}} = 1$ when $c_i \in \text{topic}(d_j)$, 0 otherwise. Here $\alpha \ge 0 \in \mathbb{R}$ is the learning rate. The category descriptor weight v_{ki} is updated as $v_{ki} + \delta(v_{ki})$, $1 \le k \le \mathcal{T}$, whenever category c_i takes part in an erroneous classification. If d_j is misclassified into c_i then $(\text{conf}_{\text{req}} - \text{conf}(d_j, \text{descr}(c_i)))$ is negative, hence the weight of co-occurring terms in d_j and c_i are reduced in the category descriptor of c_i. In the other case, if c_i is the correct but unselected category of d_j, then $(\text{conf}_{\text{req}} - \text{conf}(d_j, \text{descr}(c_i)))$ is positive, thus the weight of co-occurring terms in d_j and c_i are increased in the category descriptor of c_i.

We also experimented with a more sophisticated weight setting method where the previous momentum of the weight modifier is also taken into account in the determination of the current weight modifier. Let $\delta^{(n)}(v_{ki})$ be the weight modifier in the nth training cycle, and $\delta^{(0)}(v_{ki}) = 0$ for all $1 \le k \le \mathcal{T}$. Then the weight modifier of the next training cycle is $\delta^{(n+1)}(c_i) = \langle \delta^{(n+1)}(v_{1i}), \ldots, \delta^{(n+1)}(v_{\mathcal{T}i}) \rangle$, and its elements are calculated as

$$\delta^{(n+1)}(v_{ki}) = \alpha \cdot (\text{conf}_{\text{req}} - \text{conf}(d_j, \text{descr}(c_i)))) \cdot w_{ij} + \delta^{(n)}(v_{ki}) \cdot \beta \tag{7}$$

where $\beta \in [0, 1]$ is the momentum coefficient. The value of α and β can be uniform for all categories, or can depend on the level of the category. We experienced that at a lower value, typically 0.05..0.2 is better if the number of training documents is plentiful, i.e. higher in the hierarchy, and a higher value is favorable when only a few training documents are available for the given category, i.e. at leaf categories.

The number of nonzero weights in category descriptors increases as the training algorithm operates. In order to avoid their proliferation, we propose to set descriptor term weights to zero under a certain threshold.

The training cycle is repeated until the given maximal iteration has not been finished or the performance of the classifier does not improve significantly. We use the following optimization (or quality) function to measure inter-training effectiveness of the classifier for a document d:

$$Q(d) = \frac{\#(\text{correctly found topics of } d)}{\#(\text{total topics of } d)} \cdot \frac{1}{1 + \#(\text{incorrectly found topics of } d)}.$$

The overall Q is calculated as average of $Q(d)$ values:

$$\overline{Q} = \frac{\sum_{d \in \mathcal{D}_{\text{Train}}} Q(d)}{|\mathcal{D}_{\text{Train}}|} \tag{8}$$

The quality measure $\overline{Q}$ is more sensible to small changes in the effectiveness of the classifier than, e.g., F-measure [28] that we use to qualify the final performance of the classifier (see Section 4). Hence, it is more suitable for inter-training utilization. By setting a maximum variance value $\text{var}_{\max}$ (typically 0.95..1.00) we stop training when actual $\overline{Q}$ drops below the $\text{var}_{\max} \cdot \overline{Q}^{\text{best}}$, where $\overline{Q}^{\text{best}}$ is the best $\overline{Q}$ achieved so far during training.

4 Implementation and experimental results

4.1 Document collections

To compare our algorithm with other hierarchical classifiers we tested its effectiveness on three document corpora.

The simplest one in terms of the number of categories is document collection set up by Chuang that contains TV closed caption data mixed with web pages (15%) on the domain of professional baseball and basketball news [6] (courtesy of W. Chuang). The collection consists of 128 documents, that has been divided to 91 training and 37 test documents. The three level topic hierarchy is depicted on Figure 2. There are 11 leaf categories, therefore the average number of training documents/category is 8.273.

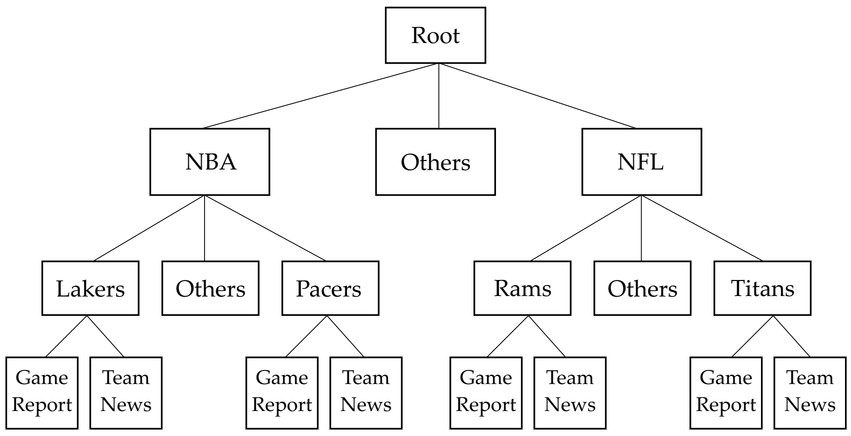

Fig. 2. Three level hierarchy on the domain of professional basketball and baseball news (redrawn from [6])

The second one is the well-known Reuters-21578 [5] newswire benchmark data collection. The collection contains 135 independent categories, i.e. originally they are not organized in hierarchy. In order to use, though, the collection as benchmark of hierarchical categorization and to prove its superiority to flat categorization several authors organized the categories into hierarchies. We make use of 3 hierarchies proposed by D'Alessio et al [7, page 10–11] (Experiments E3, E4, E5) but we also applied are our classifier to the original flat category system. We replaced the "one-of-M" relationships in all hierarchy by binary relationship. We have adopted the mode-Apté [2] split that removes all unlabelled documents resulting in total 7760 training and 3009 test documents. It means that we have not used the 1843 training and 290 test documents without topic. Some documents in Reuters collection are classified into more than one topic. The average number of categories/document 1.238. The distribution of documents among categories are quite uneven, there exist categories with several thousands training documents, and with no documents at all. Number of categories without training and test documents is 19.

The third document corpus is the 20-newsgroups data set [6]. We applied the hierarchy that was used by McCallum et al in [17] (A. McCallum, personal communication). The hierarchy has 5 topics at top level and 15 topics at the second level (see also Figure 3). All documents are assigned uniquely to one leaf-category, and each leaf-category owns 1000 documents. The documents are not divided into training and test document sets therefore we apply k-fold cross-validation approach to test the efficiency of the classifier.

4.2 Dimensionality reduction

When dealing with large document collection, the large number of terms, $|\mathcal{T}|$, can cause problem in document processing, indexing, and also in category induction. Therefore, before indexing and category induction many authors apply a pass of *dimensionality reduction* (DR) to reduce the size of $|\mathcal{T}|$ to $|\mathcal{T}'| \ll |\mathcal{T}|$ [25]. Beside that it can speed up the categorization, papers also reported that it can increase the performance of the classifier with a few percent, if only a certain subset of terms are used to represent documents (see e.g. [12,30]). In our previous experiments [26] we also found that performance can be increased slightly (less than 1%) if rare terms are disregarded, but the effect of DR on time efficiency is more significant. We applied DR by removing the least frequent terms in the overall collection. In general, we reduced $|\mathcal{T}|$

[5] The Reuters-21578 collection may be freely downloaded from `http://www.daviddlewis.com/resources/testcollections/reuters21578/`.

[6] Available from `http://www.ai.mit.edu/people/jrennie/20 newsgroups/`

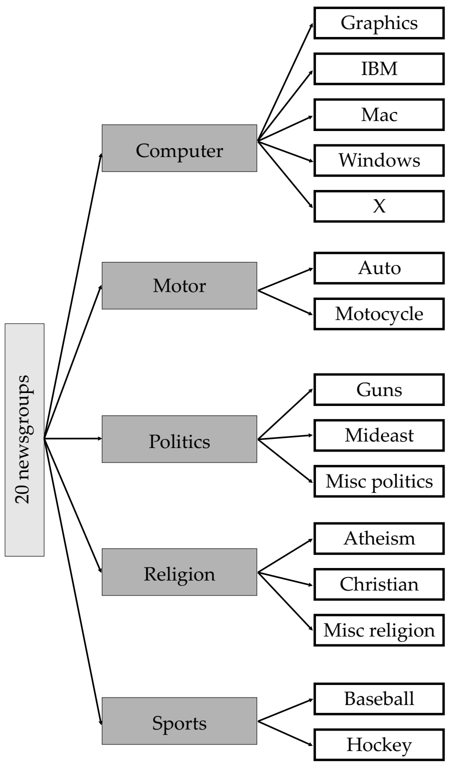

Fig. 3. Hierarchy on 20 newsgroups data set (after A. McCallum)

by disregarding terms that satisfy

$$\theta \cdot \sum_{d_j \in \mathcal{D}_{\text{Train}}} o_{kj} < \sum_{k=1}^{|\mathcal{T}|} \sum_{d_j \in \mathcal{D}_{\text{Train}}} o_{kj},$$

where integer threshold parameter θ depends on the collection, and is typically in the range $[1000, 50000]$ (kth term's total occurrence is less than $1/\theta$th part of the cumulated total occurrences of all terms). Obviously, the number of terms influences the average size of a document term weight vector (2) (if zeros are not stored), and hence the speed of classification.

4.3 Performance evaluation

In our view the quality of hierarchical classification is better if a document classified incorrectly only from an intermediate level than if it is assigned a completely incorrect topic path. In such cases when the correct leaf-category of a document is not found, but the original and the inferred topic paths partially equal the classification is not completely wrong. This view is reflected in our quality measure Q (see (8)), and we also adopt when calculating the final effectiveness of the classifier (both for training or test documents) by means of the F_1-measure. Many authors used *accuracy* or *break-even point* to measure the effectiveness of categorization. The former has been criticized due to its insensitivity [25, page 34],[32], while the latter has been recently criticized also by its proposer Lewis [15],[25, footnote 19] "since there may be no parameter setting that yields the break-even", and "to have equal recall and precision value is not necessarily desirable".

The F_β-measure originally defined by van Rijsbergen [28] is a combination of the microaveraged recall (ρ) and precision (π) values:

$$F_\beta = \frac{(\beta^2+1)\pi\rho}{\beta^2\pi+\rho}, \quad 0 \leq \beta < +\infty.$$

As many authors, we have been used F_1-measure to test the effectiveness of the classifier *on all categories*, i.e. not only on leaf-categories (but, obviously, excluding the root, as it is, by definition, not included in topic paths, see (1)).

We also tested the *training* and *classification* efficiency [8] in terms of required time (all experiences have been performed on a 1.06 GHz, 256 MB RAM PC) and its dependency from the size of the term set, $|\mathcal{T}|$.

4.4 Results

Table 1 shows the results obtained on basketball and baseball news data set. The main reasons of the high effectiveness are the small size of the topic hierarchy and the good distribution of training documents. Time requirement for training and testing (20 training loops and 1 test) is under half second regardless the size of the term set. The conf_{min} parameter is set to 0.36, $\text{var}_{\text{max}} = 0.99$. Table 2 shows the effect of DR on categorization. We varied θ and fixed all other parameters. The number of training loops was constantly 20, expect for two smallest θ values when it was 21 (300) and 38 (200). The latter explains the relatively long training time in this case. We can conclude that size of the term set does not affect significantly the performance, if it is kept over a reasonable level: when $\theta \geq 700$ the maximum variance from the best F_1-measure is just over 3%. But even when $|\mathcal{T}| = 22$, and the average number of terms in a document is 6.39 our method provides better overall categorization results then hierarchical TFIDF [6]. F_1-measure for training documents is 1 except in the case of last three rows.

Table 3 gives the results achieved with the E3, E4, E5 hierarchies [7] and with the regular flat category system. The parameters are $\text{conf}_{\text{min}} = 0.25$, $\text{var}_{\text{max}} = 0.98$. Observe that our shows the same relationship between the hierarchies as reported by D'Alessio et al: E4 hierarchy yields the best result. Wibowo and Williams [30] experienced this collection with another hierarchy [10], perhaps this is the reason why they best result is even lower, 73.74%, than that has been achieved by flat categorizers. Our method achieved remarkable results on flat category system as well. This is comparable with the best known results of flat categorizers (87.8% Weiss et al [29], committee of decision trees).

Table 4 reports on our experience with 20 newsgroup data set using the hierarchy of Figure 3. The curve of effectiveness raises rapidly with the increase of training documents (similarly as in [17]) and it becomes flat above 20%. These experiments were done with full term set (no DR has been applied to it) in order to be comparable with the results of McCallum et al. The size of term set is 55884. The magnitude of the learning rate α influences the time efficiency of the method as lower values mean slower learning and it is also reflected significantly in the elapsed time. For example in the case of 3% training document ratio, lower α values result in up to 30% slower training and a not very significant 0.2% increase in performance. Wibowo and Williams achieved better results with a fixed 60% train 40% test split on a different version of the collection when they used fewer terms for document indexing. Their best result, 90.43% by means of the hierarchical version of Rocchio classifier and 80 selected terms per documents, is 1.07% superior to ours, however these numbers cannot be directly compared due to the differences in experiment settings. In the near future we intend to investigate the effect of DR on this corpus too.

5 Application to FRT expansion

This section is devoted to describe how our method can be used for Fuzzy Relational Thesaurus (FRT) expansion. A thesaurus in an information retrieval system (IRS) can be considered as a *knowledge base* that represents the conceptual model of certain subject domain [14,18,21]. In fuzzy thesauri concepts are usually organized into a hierarchy being connected via different kinds of $[0, 1]$-weighted relations. When we utilize text categorization to aid FRT expansion we exploit the analogous hierarchical structure of *topic* and *concept* hierarchies. We adapt FRT described in details in [5]. Manual FRT expansion, performed by domain experts, is a lengthy and costly operation. We aim at speed up this process and simultaneously cut down its costs by offering a tool for domain experts for semi-automatical FRT expansion using text categorization.

Expansion process starts with a semi-automatically made FRT [5]. The concept hierarchy of FRT is transferred to a topic hierarchy simply by using concept names as category names and disregarding all relationships of FRT except the broader/narrower relationship between concepts. These relations defines the hierarchical structure of the text categorization. The next step is the build up of the document collection for training the categorizer. This is done by automatically retrieving documents from the web using a search engine on the category names (N.B. these are originally concept names). Assigning these documents as training examples to categories in the hierarchy we can train the categorizers. As it is described in Section 3 the categorizer tunes category descriptors. We make use of selected terms from category descriptors with high weights. As each category represents a concept in the FRT, the most typical terms of a category can be used to expand to corresponding concept in FRT by adding these terms as its subconcepts.

Obviously, not all selected terms of category descriptors are suitable to expand the FRT with: this task necessitates filtering in order to keep the knowledge base of FRT consistent. We therefore offer in the implementation an option for the domain expert to supervise the insertion of these terms and/or modify their weight before added permanently to the FRT reflecting his/her view on the subject domain the best. In order to maintain the quality of FRT some changes should be applied in document indexing. We allow the use of n-grams in the indexing process with $n = 3$ and do not perform stemming on the term set. Although this increases significantly the size of the term set, as a result we obtain more meaningful terms for FRT expansion. When FRT is expanded with new concepts obtained from text categorization, we can further expand and refine the FRT by starting again the expansion process with the new FRT.

6 Conclusion and further works

We proposed a new method for text categorization, which uses a iterative supervised learning method to train the classifier. We showed the effectiveness of the algorithm on three different document corpora with six topic hierarchies of different sizes. The main advantage of our algorithm is that it builds up the classifier gradually by a supervised iterative learning method, thus we can feedback the intermediate experiments to the method when training. We intend to extend the experiments with our algorithm on other larger document corpora having much more documents in the near future. We indicated another application of our text categorization method on the field of fuzzy relational thesauri: the expansion of knowledge base can be supported in a cost-effective way.

References

[1] L. Aas and L. Eikvil. Text categorisation: A survey. Raport NR 941, Norwegian Computing Center, 1999.

[2] C. Apte, F. J. Damerau, and S. M. Weiss. Automated learning of decision rules for text categorization. *ACM Trans. Information Systems*, 12(3):233–251, July 1994.

[3] K. D. Baker and A. K. McCallum. Distributional clustering of words for text classification. In *Proc. of the 21th Annual Int. ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, pages 96–103, Melbourne, Australia, 1998.

[4] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *The VLDB Journal*, 7(3):163–178, 1998.

[5] J. H. Choi, J. J. Park, J. D. Yang, and D. K. Lee. An object-based approach to managing domain specific thesauri: semiautomatic thesaurus construction and query-based browsing. Technical Report TR 98/11, Dept. of Computer Science, Chonbuk National University, 1998. `http://cs.chonbuk.ac.kr/∼jdyang/publication/techpaper.html`.

[6] W. Chuang, A. Tiyyagura, J. Yang, and G. Giuffrida. A fast algorithm for hierarchical
text classification. In *Proc. of the 2nd Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK'00)*, pages 409–418, London–Greenwich, UK, 2000. `http://www.cs.iastate.edu/∼yang/Papers/dawak00.ps`.

[7] S. D'Alessio, K. Murray, R. Schiaffino, and A. Kershenbaum. The effect of using hierarchical classifiers in text categorization. In *Proc. of 6th International Conference Recherche d'Information Assistee par Ordinateur (RIAO-00)*, pages 302–313, Paris, France, 2000. http://133.23.229.11/ ysuzuki/Proceedingsall/RIAO2000/Wednesday/26BO2.ps.

[8] S. T. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proc. of 7th ACM Int. Conf. on Information and Knowledge Management (CIKM-98)*, pages 148–155, Bethesda, MD, 1998.

[9] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[10] P. J. Hayes and S. P. Weinstein. CONSTRUE/TIS: a system for content-based indexing of a database of news stories. In A. Rappaport and R. Smith, editors, *Proc. of the 2nd Conference on Innovative Applications of Artificial Intelligence (IAAI-90)*, pages 49–66, Menlo Park, 1990. AAAI Press.

[11] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. Technical Report, University of Dortmund, Dept. of Informatics, Dortmund, Germany, 1997.

[12] D. Koller and M. Sahami. Hierarchically classifying documents using a very few words. In *International Conference on Machine Learning*, volume 14, San Mateo, CA, 1997. Morgan-Kaufmann.

[13] R. Korfhage. *Information Storage and Retrieval.* Wiley, New York, 1997.

[14] H. L. Larsen and R. R. Yager. The use of fuzzy relational thesaurus for classificatory problem solving in information retrieval and expert systems. *IEEE Trans. on Systems, Man, and Cybernetics*, 23(1), 1993.

[15] D. D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proc. of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval*, pages 37–50, Copenhagen, Denmark, 1992.

[16] D. D. Lewis and M. Ringuette. A comparison of two learning algorithms for text classification. In *Proc. of the Third Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, 1994.

[17] A. McCallum, R. Rosenfeld, T. Mitchell, and A. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proc. of ICML-98*, 1998. `http://www-2.cs.cmu.edu/~mccallum/papers/hier-icml98.ps.gz`.

[18] S. Miyamoto. *Fuzzy Sets in Information Retrieval and Cluster Analysis.* Number 4 in Theory and Decision Library D: System Theory, Knowledge Engineering and Problem Solving. Kluwer, Dordrecht, 1990.

[19] D. Mladenić and M. Grobelnik. Feature selection for classification based on text hierarchy. In *Working Notes of Learning from Web, Conference on Automated Learning and Discovery (CONALD)*, Pittsburg, USA, 1998.

[20] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proc. of the 15th National Conference on Artifical Intelligence, AAAI-98*, 1998.

[21] T. Radecki. Fuzzy set theoretical approach to document retrieval. *Information Processing and Management*, 15(5):247–259, 1979.

[22] G. Salton. *Automatic Text Processing: the Transformation, Analysis, and Retrieval of Information by Computer.* Addision-Wesley, Reading, Massachusetts, 1989.

[23] G. Salton and M. J. McGill. *An Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[24] R. E. Schapire and Y. Y. Singer. Boostexter: a boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[25] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, March 2002.

[26] D. Tikk, J. D. Yang, and S. L. Bang. Hierarchical text categorization using fuzzy relational thesaurus. Submitted to *Kybernetika.*

[27] D. Tikk, J. D. Yang, P. Baranyi, and A. Szakál. Fuzzy relational thesauri in information retrieval: automatic knowledge base expansion by means of classified textual data. In Alen Lovrenčić and Imre J. Rudas, editors, *Proc.*

of the 6th Int. Conf. on Intelligent Engineering Systems (INES 2002), pages 203–207, Opatia, Croatia, 2002.

[28] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 2nd edition, 1979. `http://www.dcs.gla.ac.uk/Keith`.

[29] S. M. Weiss, C. Apte, F. J. Damerau, D. E. Johnson, F. J. Oles, T. Goetz, and T. Hampp. Maximizing text-mining performance. *IEEE Intelligent Systems*, 14(4):2–8, July/August 1999.

[30] W. Wibovo and H. E. Williams. Simple and accurate feature selection for hierarchical categorisation. In *Proc. of the 2002 ACM symposium on Document engineering*, pages 111–118, McLean, Virginia, USA, 2002.

[31] E. Wiener, J. O. Pedersen, and A. S. Weigend. A neural network approach to topic spotting. In *Proc. of the 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 22–34, 1993.

[32] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1–2):69–90, 1999. `http://citeseer.nj.nec.com/yang97evaluation.html`.

Table 1
Results on basketball and baseball news data set [6] (F_1-measure)

	Training				Test			
method	at depth			over-	at depth			over-
	1	2	3	all	1	2	3	all
TFIDF ([6])	96.0	91.0	90.0	92.6	84.0	81.0	58.0	75.6[†]
Our method	100	100	100	100	98.0	93.5	80.8	91.6

[†] Our calculation, based on the published results in [6].

Table 2
Size of term set vs. overall F_1-measure , elapsed time, average document vector size, elapsed time (all other parameters are fixed)

$\|\mathcal{T}\|$	θ	avg. doc size	time/doc (ms)	F_1
2600	∞	62.64	4.5	89.95
1263	10000	52.19	3.9	90.05
844	4000	46.33	3.5	90.05
653	3200	42.52	3.3	90.62
519	2500	39.04	2.9	89.69
419	2000	85.69	2.6	90.62
348	1800	33.05	2.4	91.10
298	**1500**	**30.91**	**2.3**	**91.58**
258	1300	29.09	2.3	89.79
211	1000	26.49	2.1	87.56
149	800	22.37	2.0	91.19
125	700	20.50	2.0	89.12
75	500	15.36	1.9	83.94
41	300	10.17	1.6	83.94
22	200	6.39	2.4	80.21

Table 3
Results with E3–E5 and flat hierarchy on Reuters-21578 collection. Elapsed time is meant including indicated training loops and one test pass.

	Our method						D'Alessio et al [7]		
hier-archy	F_1	π	ρ	# iter.	avg. doc size	time/doc (ms)	F_1	π	ρ
E3	91.19	90.97	91.42	10	5.8	43.32	79.8	81.4	78.3
E4	93.61	93.27	93.96	13	7.9	43.67	82.8	85.9	79.9
E5	92.39	91.78	93.02	10	4.9	43.67	82.5	86.4	79.0
flat	88.15	88.00	88.30	14	22	44.88	78.9	80.3	77.6

Table 4
Results on the 20 newsgroups data set with hierarchy depicted on Figure 3. Our results are the average of 10 run with randomly selected training documents. Bracketed numbers show the value of learning rate α (see (6)) at the first and second level.

% documents used for training	method (performance measure)	
	Our method (F_1-measure)	Hierarchical Naïve Bayes with shrinkage [17] (accuracy[†])
1	68.58% (1.00; 1.10)	$\approx$ 52.0
2	74.17% (0.40; 0.80)	$\approx$ 58.0
3	77.06% (0.06; 0.12)	$\approx$ 62.0
5	80.16% (0.06; 0.12)	$\approx$ 68.0
7	81.90% (0.05; 0.10)	$\approx$ 72.0
10	83.27% (0.05; 0.10)	$\approx$ 75.0
20	86.31% (0.05; 0.10)	$\approx$ 77.5
30	87.44% (0.05; 0.10)	$\approx$ 81.0
40	88.27% (0.05; 0.10)	$\approx$ 82.5
50	88.91% (0.05; 0.10)	$\approx$ 83.0
60	89.34% (0.05; 0.10)	$\approx$ 83.5
70	89.54% (0.05; 0.10)	$\approx$ 84.0

[†] Estimated values based on [17, Figure 3].