Hierarchical text categorization using fuzzy relational thesaurus*

Domonkos Tikk[1,2,3†], Jae Dong Yang[1] and Sun Lee Bang[1]

[1] Dept. of Computer Science, Chonbuk National University
Chonju 561–756, Korea
e-mails: {jdyang,sibang}@cs.chonbuk.ac.kr

[2] Dept. of Telecommunications & Telematics,
Budapest University of Technology and Economics,
1117 Budapest, Magyar Tudósok Körútja 2., Hungary,
e-mail: tikk@ttt.bme.hu

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

Abstract

Text categorization is the classification to assign a text document to an appropriate category in a predefined set of categories. We present a new approach for the text categorization by means of Fuzzy Relational Thesaurus (FRT). FRT is a multilevel category system that stores and maintains adaptive local dictionary for each category. The goal of our approach is twofold; to develop a reliable text categorization method on a certain subject domain, and to expand the initial FRT by automatically added terms, thereby obtaining an incrementally defined knowledge base of the domain. We implemented the categorization algorithm and compared it with some other hierarchical classifiers. Experimental results have been shown that our algorithm outperforms its rivals on all document corpora investigated.

Index terms

Text mining; Knowledge base management; Multi-level categorization; Hierarchical text categorization.

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

*This research was mainly done while the first author was visiting and supported by the Chonbuk National University, Korea. This work was also funded by the Hungarian Scientific Research Fund (OTKA) Grant No. D034614 and by the Hungarian Ministry of Education Grant No. FKFP 0180/2001.

†Corresponding author

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, 15, 27, 28, 29]. 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, multi-label problems have received very little attention [1].

To assign documents to categories, text categorization methods usually employ *dictionaries* consisting of words extracted from training documents. The assignment is made based on frequencies of occurrence of dictionary words in a document. While conventional methods employ a large *global dictionary* [7, 11, 29], *local dictionaries* for each category [2], or *pooled local dictionary* [27], our method adopts a combined dictionary. It uses local dictionaries for each category that are incrementally expanded from the global dictionary during training.

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[1], 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[2]. Other real-world applications also often pose problems with multilevel 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 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.

We now describe a multilevel text categorizer based on Fuzzy Relational Thesaurus (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, 22]. In fuzzy thesauri concepts are usually organized into a hierarchy being connected via different kinds of

[1] In general hierarchy is considered to be an acyclic digraph; in this paper we restrict somewhat this definition, see Section 3.1.

[2] `http://www.yahoo.com`, `http://www.ibm.com/patents`

$[0, 1]$-weighted relations. In our approach FRT serves as implementation of topic hierarchies. Therefore, we call concepts of FRT in this paper topics or categories emphasizing the link between *concept hierarchy* of an FRT and *topic hierarchy* of a taxonomy. FRT stores and maintains local dictionaries consisting of descriptive terms of a category. Terms can be words or n-grams (sequence of up to n words). Local dictionaries, or more simply: *descriptors*, are incrementally built up when training FRT for categorization.

To exemplify the classification problem and terminology of our approach consider the subject domain of *electronic appliances*. A part of the taxonomy of the domain is depicted in Figure 1. Further, let us consider a document classified into topic `Laser Printer`. The topic path of the document is `CPD` $\rightarrow$ `Printer` $\rightarrow$ `Laser Printer`. The document contains the following descriptive terms: Copier (2 occurrences), FAX (3), RAM (1), Printer (2), Scanner (3), Modem (2). A document is classified based on its descriptive terms w.r.t. categories. Since initially the size of descriptors is small (they typically contains only the topic's name), the determined category may often be incorrect. Therefore we add the most typical terms of categories selected from training documents to descriptors in order to improve the the effectiveness of the classifier. Details are described in Section 3.

Our aim is twofold: Primarily, we intend to develop a hierarchical text categorization algorithm using the hierarchical structured FRT, which performs well. Secondly, we want to expand the FRT, being created manually or semi-automatically, using the trained descriptors in an automatic way to enhance its descriptive power as a knowledge base. Obviously, not all terms of the descriptors are suitable to expand the FRT with: this task necessitates filtering on descriptor elements in order to keep the knowledge base consistent. We therefore offer in the implementation an option for the user maintaining the FRT (domain expert) to supervise and/or modify these terms before added permanently to the FRT, reflecting his/her view on the subject domain the best. In this paper we concentrate only on the categorization task.

The paper is organized as follows. Section 2 reviews the related works of text categorization. Section 3 is the main part of this paper where FRT and its application in categorization is described in details. Section 4 shows experimental results and conclusion follows in Section 5.

2 Related works

Numerous statistical classification and machine learning techniques have been applied to text categorization. They include nearest neighbor classifiers (KNN) [29], regression models [29], voted classification [27], Bayesian classifiers [15], decision trees [15], Support Vector Machines (SVM) [11], information-theoretic approaches (e.g. distributional clustering) [3] and neural networks [28]. For comparative studies see [1, 25]. Usually these techniques are compared on a standardized collection of documents, such as the Reuters-21578 corpus. Among them some version of KNN, SVM and voted classification provide the best results, achieving around 86.3–87.8 percentage for break-even points (results may somewhat vary at different authors). The overall greatest break-even point, 87.8, was attained by Weiss et al [27]. 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. [10]. In [12] the authors focused on the reduction of local dictionaries (also called *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 (see also Section 4 and Tables 1 and 2) shows that hierarchical classifiers outperform flat ones when the number of features is small (less than 100). Their approach was criticized in e.g. [16], 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, 16, 20].

As alternative approaches, hierarchical text categorization was combined in many works with feature subset selection. The feature subset selection improved classification accuracy, reduced 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 [16]. 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, 23] (using tf×idf weighting, see (5)) 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 a difficulty due to the different text corpora and their provisionality they are applied to. We return to this problem in Section 4.

3 The proposed method

The core idea of the FRT based categorization is the training algorithm that adds new terms to the descriptors with certain weights, and modifies the weight of terms if necessary. We start from a small FRT manually created by a domain expert (see [5]), which describes the topic hierarchy the documents have to be categorized into. This FRT should contain the topic hierarchy as its subgraph (to be detailed later) with possible empty descriptors, but may also contain some other categories. The initial descriptors typically just a few terms, or even none.

We now briefly describe the training procedure. Primary, we use the descriptors of topics in the FRT to categorize a document. When this procedure fails due to, e.g., the small size of descriptors, we insert new terms into the descriptor of the correct category from the *expansion set* of the document. The expansion set is a frequency ordered set of terms of the document belonging to the given category. Its size is controlled by threshold parameters. When FRT

determines an incorrect topic, those terms in its descriptor which generated this choice are penalized by weakening their weights. The training algorithm is executed in an iterative way, and it ends when the performance cannot be further improved significantly. See the block diagram of Figure 2 for an overview and details in Subsection 3.4 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 vector space model, notation and terminology. Subsection 3.2 focuses on descriptors and expansion sets. Subsection 3.3 presents the core of our algorithm for document classification based on FRT, and finally Subsection 3.4 includes the training of FRT in detail.

3.1 Definitions

3.1.1 Taxonomy and FRT

Let $\mathcal{C}$ be the fixed finite set of categories organized in a *topic hierarchy*. We refer to a hierarchy as a set of disjoint acyclic digraphs connected under a root. The root does not represent any category hence it does not take part in the categorization. An acyclic digraph describes topics under one top level category. A node can have more than one parent, where parents should be in the same subgraph due to the disjointness condition. See below an example of such multiple parentcraft.

Each document d is classified into a leaf category of the hierarchy. 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 leaf to the root.

We allow multiple parentcraft because topics deeper in the hierarchy can have strong relations, especially when the taxonomy describes a relative small subject domain as in our example. On the example of Figure 4 the topic "Cassette MP3 Player" belongs to both of the categories "MP3 Player" and "Cassette".

Each topic $c \in \mathcal{C}$ is assigned a *level* or *depth* in the taxonomy that is defined recursively by the function $\text{level} : \mathcal{C} \to \mathbb{N}$ as

$$\text{level}(c) = \begin{cases} 0, & \text{if } c \text{ is the root} \\ \min\limits_{c' \text{ is parent of } c} (\text{level}(c')) + 1, & \text{otherwise} \end{cases} \tag{1}$$

The depth of a taxonomy is defined as the level of the deepest category:

$$\text{depth}(\mathcal{C}) = \max_{c \in \mathcal{C}} \text{level}(c). \tag{2}$$

The topic hierarchy just described offers a straightforward way to connect the categorization purpose and fuzzy relational thesauri [14] having a hierarchically organized structure. The relations between FRT elements are weighted by real number of $[0, 1]$ interval, or alternatively, we can say that relations are fuzzy. We adapt FRT described in details in [5] for text categorization by disregarding the various type of relationships, i.e. we use uniquely the broader/narrower concept (in our application: topic) relationship.

As we mentioned in the Introduction, in our approach FRT serves as implementation of topic hierarchy. There are two cases. First, if an FRT is created solely as an implementation

basis of the categorization purpose, its graph is then identical with the taxonomy. If an already existing FRT is used for categorization, we require that FRT, being also an acyclic digraph, should contain the taxonomy as its isomorphic and level invariant subgraph (see also Figure 3). In this case the only this subgraph of FRT takes part in the categorization. Descriptors of other nodes are cleared, and cannot be augmented because no training documents can belong the them. Due to this structural identity of taxonomy and FRT, we shall refer to an element of the FRT hierarchy as *topic* or *category*.

3.1.2 Vector space model

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.

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 increases dramatically the storage need of the model, and as it was reported in [2, 9] 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*

$$d_j = (w_{1j}, \ldots, w_{|\mathcal{T}|j}), \tag{3}$$

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 $\mathcal{T}$. The term set is often called *universal dictionary* as well.

In certain settings of our method we also use the *occurrence* vector in the characterization of document d_j:

$$\text{occur}(d_j) = \langle o_{1j}, \ldots, o_{|\mathcal{T}|j} \rangle, \tag{4}$$

where $o_{kj} \in \mathbb{N}$ determines the number of occurrence of kth term in document d_j.

There are numerous possible weighting schemes in the literature to determine the values of term weights w_{kj}. The best and most sophisticated method is the entropy weighting, which was found to outmatch 6 others in [8], but we apply the most popular tf×idf weighting [24], 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:

$$w_{kj} = o_{kj} \cdot \log\left(\frac{N}{n_k}\right), \tag{5}$$

Term vectors (3) are normalized before training.

The document d_j classified into a leaf category c also belongs to its parent categories, i.e. it belongs to all categories between c and the root along a topic path. Formally,

$$\text{topic}(d_j) = \{c_1, \ldots, c_q \in \mathcal{C} | \, \text{level}(c_i) \geq \text{level}(c_j) \geq 1, \text{ when } i < j\} \tag{6}$$

determines the set of topics d_j belongs to along the topic path from the deepest to the highest. Note that c_q is the top level category of that subgraph where d_j is classified into, i.e. we disregard the root. Because multiple parentcraft is allowed, it can happen that a topic path contains several categories of the same (intermediate) level.

3.2 Descriptors and expansion sets

FRT based classification works based on the matching between vectors representing documents and categories. We represent categories analogously as documents. It is a vector of *descriptor term weights*

$$\text{descr}(c_i) = (v_{1i}, \ldots, v_{|\mathcal{T}|i}), \quad c_i \in \mathcal{C} \tag{7}$$

where weights $0 \leq v_{1i} \leq 1$ are set during training. The weight of initial descriptors (given by the domain expert) is 1. All other weights are initialized as 0. The descriptor of a category can be interpreted as the prototype of a document belonging to it.

Usually the initial number of descriptive terms (often being zero) is not sufficient for an efficient classifier. In order to fill up the descriptor during training, we create *expansion sets* in the preprocession phase. During training when categorization fails, terms from expansion sets are added to descriptors to increase the efficiency of the classification.

We create for each training document $d \in \mathcal{D}_{\text{Train}}$ and for each topic in $c \in \text{topic}(d)$ an expansion set E_c^d. It is a frequency ordered set of terms of the document d characterizing category c. Each term of d is assigned a cumulated value, which is the sum of the appropriate term weights of the (at most) k-nearest neighbors of d belonging to the same category c. Note that k may be greater than the number of documents in c, whence less than k documents are considered. By this selection mechanism we can balance the number of considered documents if a topic that contains a large number of training documents.

$E_c^{d_j}$ contains the elements of $\mathcal{T}$ in descending order according to the cumulated term weights of d_j's at most k nearest neighbors calculated as

$$p_i = \begin{cases} 0, & \text{if } w_{ij} = 0 \\ w_{ij} + \sum_{\ell \in L} w_{i\ell}, & \text{if } w_{ij} > 0 \end{cases}. \tag{8}$$

This value is assigned to the ith term $1 \leq i \leq |\mathcal{T}|$. Here the set L contains the index of the at most k nearest neighbors of document d_j in category c, $|L| \leq k$. Hence,

$$E_c^{d_j} = \{(i_n, p_{i_n}) | 1 \leq i_n \leq |\mathcal{T}|, i_n < i_m \text{ if } p_{i_n} > p_{i_m}, \forall 1 \leq n, m \leq |\mathcal{T}|; n \neq m\} \tag{9}$$

The distance between documents is calculated by the cosine measure.

Note that the concept of the expansion set is different from that of the descriptor. First, because the former is assigned to a category of a document while the latter assigned to a

category. Second, the former is an ordered set of terms of a document whereas the latter is a set of terms characterizing an entire topic. The topic descriptors are augmented during training based on expansion sets.

3.3 Classification by means of FRT

When classifying a document $d \in \mathcal{D}$ by means of the FRT the term vector representing d (3) is compared to topic descriptors (7). 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. This greedy type algorithm ends when a leaf category is found. McCallum [16] criticized the greedy topic selection method because it requires high accuracy at internal (non-leaf) nodes. In our experiments (see Section 4) this algorithm performs very well, but in our future works we plan to consider other algorithms where a node can "reject" a document and send it back upward the hierarchy for re-classification.

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 similarity 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 similarity measure. We carried out experiences with three similarity measures:

1. The simplest is binary comparison that calculates the number of terms that mutually occurs in the descriptor and in the document:

$$s_1(d_j, \mathrm{descr}(c_i)) = \sum_{k=1}^{|\mathcal{T}|} \mathrm{sign}(w_{kj} \cdot v_{ki}), \tag{10}$$

where sign is the signum function.

2. The second is the unnormalized cosine similarity measure that calculates the value as the sum of products of document and descriptor term weights:

$$s_2(d_j, \mathrm{descr}(c_i)) = \sum_{k=1}^{|\mathcal{T}|} w_{kj} \cdot v_{ki}. \tag{11}$$

3. The third similarity measure also takes into account the occurrence vector (4):

$$s_3(d_j, \mathrm{descr}(c_i)) = \sum_{k=1}^{|\mathcal{T}|} w_{kj} \cdot o_{kj} \cdot v_{ki}. \tag{12}$$

(As $\mathrm{occur}(d_j)$ depends on d_j the operand set does not change.) This measure strongly emphasizes multiple occurring terms of the document.

After matching vectors of document term weights and category descriptor weights against each other, we control the selection of the best category by a minimum conformity parameter $\min_{\mathrm{conf}} \in [0, 1]$, i.e. the greedy selection algorithm continues when

$$s(d_j, \mathrm{descr}(c_{\mathrm{best}})) \geq \min_{\mathrm{conf}}$$

satisfied, where s is an arbitrary similarity measure and c_{best} is the best category at the given level.

Example: We show on a simplified example the category selection method using the above three similarity measures at top level of the taxonomy introduced on Figure 1. Let document $d = (0, 0.6, 0.4, 0, 0.1, 0.656, 0.2, 0)$, $\text{descr}(\text{CPD}) = (0, 0.8, 0.6, 0, 0, 0, 0, 0)$, $\text{descr}(\text{OCA}) = (0, 0, 0, 0, 0, 0.8, 0, 0.6)$, $\text{descr}(\text{CC}) = (0.1, 0.1, 0, 0, 0.8, 0, 0.5, 0.3)$

With (10) we obtain $s_1(d, \text{CPD}) = 2$, $s_1(d, \text{OCA}) = 1$, $s_1(d, \text{CC}) = 3$, and "Computer Components" (CC) is selected. With (11) we get $s_2(d, \text{CPD}) = 0.72$, $s_2(d, \text{OCA}) = 0.5248$, $s_2(d, \text{CC}) = 0.24$ and "Computer Peripheral Devices" (CPD) is selected. If we have also $\text{occur}(d) = (0, 8, 5, 0, 1, 10, 2, 0)$ given then with (12) we obtain $s_3(d, \text{CPD}) = 5.04$, $s_1(d, \text{OCA}) = 5.248$, $s_3(d, \text{CC}) = 0.76$, and "Office Communication Appliances" (OCA) is chosen. Although this is only an explanatory example and the weight values are not from a real setting, it shows that the selection of similarity method is crucial concerning the effectiveness of the classification.

3.4 Updating the knowledge base: the training algorithm

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.

We modify descriptor term weights in two cases:

1. Case: FRT is unable to find the correct category c of a document d.

   To alleviate this type of error we raise the weight of those terms in the $\text{descr}(c)$ that characterizes best the link between c and d, the terms of expansion set E_c^d (see Section 3.2). We take a prefix of E_c^d and select only its (p_1, α_1)-level set [14, p. 37], i.e the set that contains the at most first p_1 terms with weight equal to or higher than α_1 ($p_1 \in \mathbb{N}$, $\alpha_1 \in [0, 1]$ adjustable parameters). The descriptor term weights corresponding to the selected terms are set to the p_i defined in (9). Then expansion set E_c^d is updated by removing its (p_1, α_1)-prefix. The parameters p_1 and α_1 can be redefined in each training cycle.

   Example: In the first training cycle the $(5, 0.1)$-level set of the expansion set assigned to our sample document and category *Laser Printer* consists of terms: `max print cm dpi paper`. In the second training cycle: `page comm monochrom postscript plain` (stemmed words).

2. Case: The category c determined by FRT is incorrect.

   We handle this type of error by modifying certain descriptor term weights of category c. Term weights of c having nonzero value in the term vector of document d are multiplied by a factor α_2 ($\alpha_2 \in (0, 1) \subset \mathbb{R}$ adjustable parameter). The value of α_2 controls the strength of penalization.

The size of the descriptors grows as more and more terms have nonzero weights. 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 F_1-measure [26] to check the effectiveness of the classifiers on training documents. By setting a maximum variance value $\max_{\text{var}}$ (typically 0.95..1.00) we stop training when actual F_1 drops below the $\max_{\text{var}} \cdot F_1^{\text{best}}$, where F_1^{best} is the best F_1 achieved so far during training.

4 Implementation and experimental results

4.1 Document collections

Because there is no standard document corpus particularly for multi-class and multi-label text classification test, we decided in the initial phase of our project to collect web documents in the domain of *electronic appliances* (EA).[3] (Later we performed tests on other corpora, see later). We collected 328 documents. There are $|\mathcal{T}| = 5713$ terms (size of the global dictionary) after stemming and removal of function words. FRT was created by a semi-automatic thesaurus construction software [5]. The depth of the taxonomy, depth($\mathcal{C}_{\text{EA}}$), is 3.

The collected documents were classified into the following six top level topics: Audio, Computer, Computer Components, Computer Peripheral Device, House-Hold Appliances, Office Communications Appliances. The number of topics were 31 and 58 at subsequent levels. All documents were classified into lowest level categories, hence the average number of documents/category were 5.655. Each category had at least one document. Documents were distributed evenly among the subgraphs of top level categories, except the Computer Components topic which had 178 documents. We have been applied the k-fold cross-validation approach [17, p. 146] to divide the document corpus into training and test documents.

To compare our algorithm with other hierarchical classifiers we tested its effectiveness on two other corpora: the TV closed caption data [6] (courtesy of W. Chuang) and on some subsets of Reuters-21578[4] database. While in the former case the topic hierarchy was ready [6, Figure 8.], in the latter case we used the two taxonomies introduced in [12]. As Reuters collection is not created to be a benchmark database for hierarchical classification these taxonomies deals with two subsets of categories of the entire collection organized in simple hierarchies depicted on Tables 1 and 2. We could not get from Koller and Sahami the original setting of training/testing documents they had used for the experiments in [12], so we used all documents from the specified categories according to the "ModApté" split. As in this case a document can be attached to more than one leaf category, the total number of documents are less than the sum of training and testing documents.

[3]The document collection is available online at `http://www.mft.hu/publications/tikk/DocColl.zip`.

[4]The Reuters-21578 collection may be freely downloaded from `http://www.research.att.com/~lewis/reuters21578.html`.

4.2 Performance evaluation

We have been used the F_1-measure of microaveraged recall (ρ) and precision (π) to test the effectiveness of the classifier:

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

4.3 Results

The results achieved on the EA-collection are shown in Table 3. We have been applied all three similarity measures defined in Subsection 3.3, and we found that (11) and (12) provide close to similar results. Therefore, for simplicity, we carried out all of the presented experiments by using the simple cosine similarity measure (11). We applied and k-fold cross-validation approach with k values $(2, 5, 10, 20, 50, 100)$ to separate training and test documents. Obviously results improve considerably as the value of k increases. As the average number of documents/category is very low (there are categories with only 1 document), under certain settings there are categories with no training documents. This is reflected also in the individual test results: e.g. when $k = 10$ the lowest F_1-measure among the ten averaged test results is: 0.629; the highest is: 0.852. As for other parameters, we fixed $p_1 = 5$, $\alpha_1 = 0.01$, $\alpha_2 = 0.8$, $\min_{\text{conf}} = 0.48$ and $\max_{\text{var}} = 0.99$.

We tested the *training* and *classification* efficiency [9] 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 global dictionary, $|\mathcal{T}|$. We have been modified the size of the global dictionary by removing the least frequent terms in the overall collection. The total number of 5713 terms is reduced to 3513, if terms occurring only once in the whole collection are removed. More generally, we can reduce the size of the global dictionary by disregarding terms that satisfy

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

where integer threshold parameter θ is typically in the range $[1000, 50000]$ (ith 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 (3) (if zeros are not stored), and hence the speed of classification. Table 4 shows the required time for a 10-fold cross-validation approach as a function of the size of the global dictionary. We can conclude that size of the global dictionary does not affect significantly the performance, if it is kept over a reasonable level.

Table 5 shows the results obtained on the TV closed data set. This data set consists of only 17 categories and 91 training and 37 test documents, i.e. the latter is 30% of the collection. There are 11 leaf categories, therefore the average number of training documents/category 8.273. The main reasons of the high effectiveness are the small size of the topic hierarchy and the better distribution of training documents. Time requirement for training and testing (20 training loops and 1 test) is under 0.3 sec regardless the size of the global dictionary. The $\min_{\text{conf}}$ parameter is set to 0.35, others remain unchanged.

Table 6 gives the results achieved with the Hier1 and Hier2 taxonomies on the Reuters-21758 corpus. We indicated the corresponding results from [12] as a reference despite the fact that our experiments have been achieved with a different settings, because (1) we could not get the original document setting from the authors, (2) we used also such documents that are pre-classified to more than one categories (3) they used *accuracy* to measure the effectiveness of the method, that has been criticized by many authors due to its insensitivity [25, page 34],[29]. The results are very good due to the small number of categories and the large number of documents. The $\min_{\text{conf}}$ parameter is set to 0.2, $p_1 = 8$, $\alpha_1 = 0.05$, others remain unchanged.

5 Conclusion and further works

We proposed a new method for text categorization, which uses FRT to support the classification task. We showed the effectiveness of the algorithm on three different document corpora with four 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.

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.

[7] I. Dagan, Y. Karov, and D. Roth. Mistake-driven learning in text categorization. In Claire Cardie and Ralph Weischedel, editors, *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 55–63. Association for Computational Linguistics, Somerset, New Jersey, 1997.

[8] S. T. Dumais. Improving the retrieval information from external sources. *Behaviour Research Methods, Instruments and Computers*, 23(2):229–236, 1991.

[9] 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.

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

[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 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.

[16] 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`.

[17] T. M. Mitchell. *Machine Learning*. McGraw Hill, New York, NY, 1996.

[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)*, 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] E. H. Ruspini, P. P. Bonissone, and W. Pedrycz (eds.). *Handbook of Fuzzy Computation*. Oxford University Press and Institute of Physics Publishing, Bristol and Philadelphia, 1998.

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

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

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

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

[27] 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.

[28] 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.

[29] 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`.

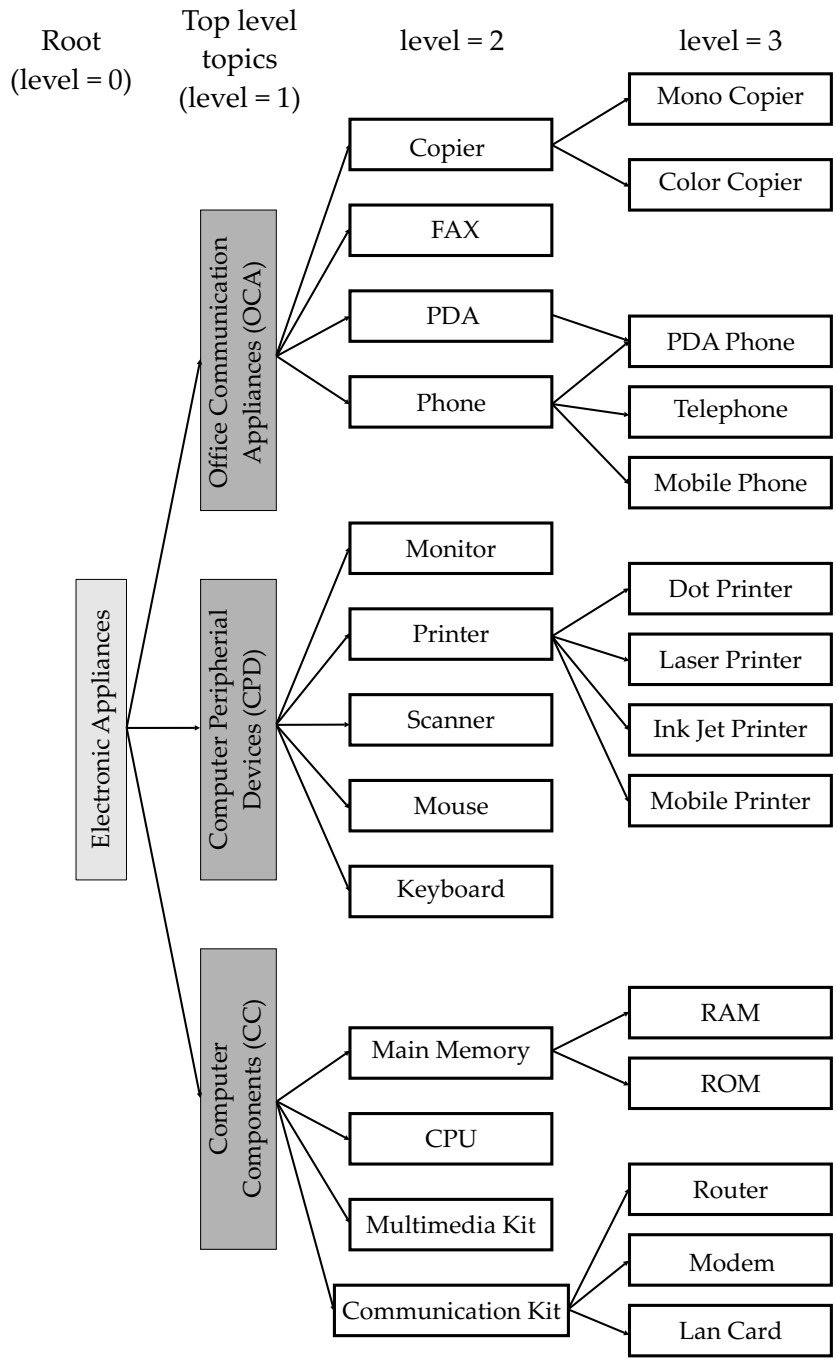

Figure 1: A part of the taxonomy Electronic Appliances. The root topic (subject domain name) is denoted by light gray, top level topics by darker gray boxes

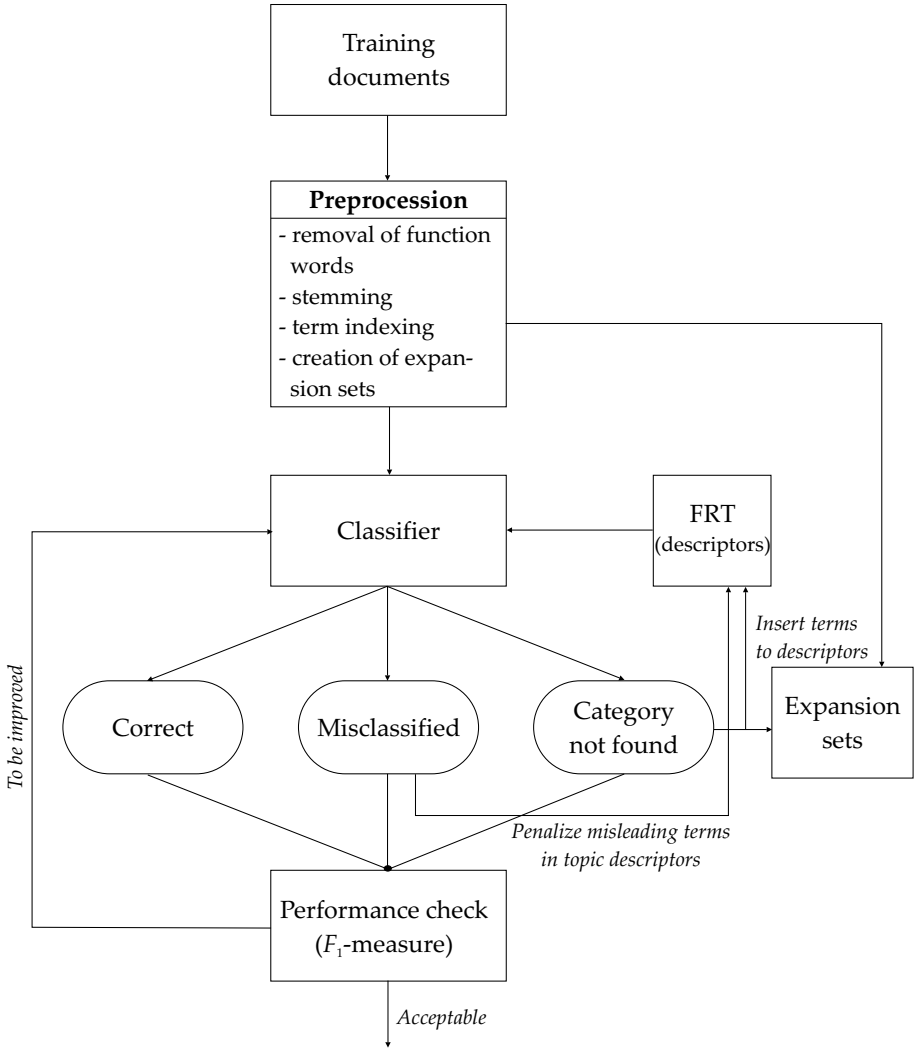

Figure 2: The flowchart of the training algorithm

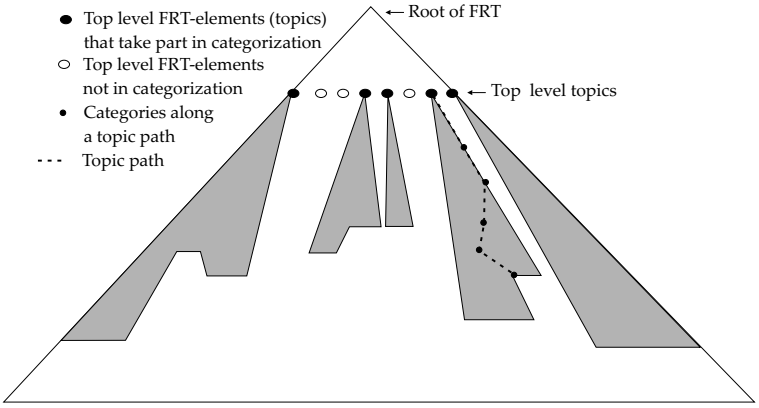

Figure 3: Taxonomy vs. FRT hierarchy: the gray area indicates the subset of FRT hierarchy that takes part in the categorization

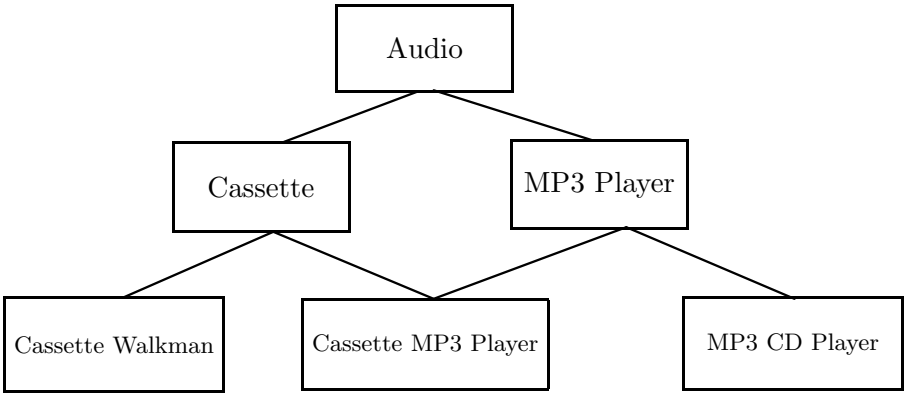

Figure 4: An example of multiple parentcraft from the Electronic Appliances taxonomy. The category "Cassette MP3 Player" is linked to "MP3 Player" and "Cassette".

Table 1: Description of Hier1 data set on Reuters-21758 (after [12])

major topics	minor topics	data set size	
		training	testing
grain business	corn	182	56
	wheat	212	70
money effects	dlr	131	44
	interest	347	131
crude oil	nat-gas	75	30
	ship	196	89
Total		1068	392

Table 2: Description of Hier2 data set on Reuters-21758 (after [12])

major topics	minor topics	data set size	
		training	testing
business	acq[†]	1657	721
	earn	2877	1087
veg. oil business	oilseed	124	47
	palmoil	30	10
Total		4661	1857

[†] We substituted `c-bonds` by `acq` because category `c-bonds` were removed from Reuters-21578.

Table 3: Categorization results on EA document set. k refers to the appropriate value of k-fold cross-validation approach. F_1-measures are averaged over k tests, while in case of precision and recall values the range of worst and best result is given.

k	Training set	Test set		
	F_1	F_1	π	ρ
10	0.999	0.770	0.774..0.872	0.713..0.837
100	0.998	0.873	0.625..1.000	0.714..1.000
50	0.998	0.860	0.706..1.000	0.733..1.000
20	0.998	0.823	0.737..0.950	0.690..0.905
5	1.000	0.720	0.744..0.820	0.639..0.699
2	1.000	0.535	0.612..0.779	0.375..0.554

Table 4: Size of global dictionary vs. elapsed time (includes 10 training runs and 10 tests), average document vector size, average of test results in case of 10-fold cross-validation approach (all other parameters are fixed)

Size of dictionary	θ	average number of term/document	elapsed time (s)	aver. no. of training cycles	F_1
5713	∞	117.84	51.71	25.1	0.762
3513	50000	111.14	45.20	25.0	0.761
2245	20000	103.51	39.51	25.0	0.761
1558	10000	95.65	35.02	25.3	0.762
1027	5000	84.83	30.88	25.9	0.767
869	4000	80.05	29.19	26.2	0.767
698	3200	73.73	27.10	26.8	0.770
552	2500	67.12	24.76	27.7	0.759
450	2000	60.84	22.40	28.0	0.759
213	1000	41.00	19.30	36.2	0.715

Table 5: Results on the TV closed caption data set [6]

	Training			Test		
method	at depth			at depth		
	1	2	3	1	2	3
Chuang et al [6]	0.96	0.91	0.90	0.84	0.81	0.58
FRT	1.00	1.00	1.00	1.00	0.95	0.87

Table 6: Results on the Hier1 and Hier2 subsets of Reuters-21578 collection

taxonomy	best result from [12]	FRT		
	accuracy	F_1	π	ρ
Hier1	0.940	0.9606	0.9581	0.9630
Hier2	0.909	0.9937	0.9935	0.9938