Text categorization with hierarhical category structure

Domonkos Tikk[*], Jae Dong Yang[**], György Biró[***], Leila Muresan[*]

* Department of Telecommunications & Telematics, Budapest University of Technology and Economics, 1117 Budapest, Magyar Tudósok Körútja 2., Hungary

** Department of Computer Science, Chonbuk National University, Chonju 561-756, Korea

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

Abstract

In this paper we describe an adaptive text categorization algorithm that is able to learn hierarchical category structures. This work was first initiated by a knowledge engineering problem. We developed a tool that provides domain engineers with a facility to create fuzzy relational thesauri (FRT) describing subject domains. Fuzzy thesauri have the ability to describe a specific domain by hierarchically organized significant words (concepts or instances) and their relationships. Creation of such structures is usually quite costly in terms of time and hence money. To fasten this procedure we aimed at providing keywords to each node of a hierarchy. These keywords being collected from categorized documents are very useful as candidates to expand FRT, because domain engineers can shorten their search time significantly when prospective candidates are offered.

For this purpose we developed a hierarchical text categorization algorithm that used the existing FRT as starting point. Selected FRT nodes are considered to be categories. The proposed method is basically a learning algorithm that composes of two phases: categorization and training. When doing categorization the system infers the category of documents, while during training the categorization ability of the system is improved based on the previous errors by enlarging the set of descriptive words of categories.

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 categorization, which aims at the automatic classification of electronic documents.

Text categorization is the classification to assign a document to an appropriate category in a predefined set of categories.

Traditionally, the 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 (see e.g. [1, 2, 3, 4]). Some of the approaches proceed very well on linear category systems, but they are inappropriate to handle to case of multilevel categories. However, real-world applications 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.

This paper proposes a new method, which classifies documents on the basis of a *fuzzy relational thesaurus* (FRT) [5]. For the categorization purpose, FRT is used to *describe* and *structure* the category set. Text categorization based on FRT is especially useful when the category set is organized in *taxonomy*, i.e. it has multilevel structure.

To assign documents to categories, text categorization methods usually employ *dictionaries*, each including a set of words extracted from training documents. The assignment is made based on frequencies of occurrence of dictionary words in a document. Conventional methods use either a large *global dictionary* [3, 4], *local dictionaries* for each category [6] or *pooled local dictionary* [2]. Our method uses two kinds of dictionaries: a global dictionary for creation of so-called *expansion sets*, and local dictionaries consisting of typical words of the corresponding categories. Each document is endowed with as many expansion sets, as many categories the document is originally assigned to. Since an expansion set captures keywords characterizing the category of a document to be classified, deciding which new terms are to be inserted into FRT wholly depends on the set.

In the remaining part of the paper we introduce our FRT based text categorization method. Its main task is to shows good performance on documents belonging to a certain subject domain that is described by the FRT implementing the category structure.

The core idea of the FRT based categorization is the training algorithm that assigns weighted words (or terms) to the categories (implemented by the FRT), and modifies weights of (word, category) and (category, category) pairs if necessary. We start from a relatively small FRT (termed basic FRT) manually created by a domain expert, which typically contains a few hundreds of terms. We assume that the basic FRT contains the category names at concept level (see [7]).

We now briefly describe the training of the FRT. Primary, we use the FRT to categorize a document. When this procedure fails due to, e.g., the small amount of terms in the basic FRT, we use the result of an alternative (statistical) text categorization approach in order to train FRT, and then to correct the categorization error. Based on the original category or categories, and on some additional information (see details later) obtained from the statistical approach, such as, e.g., the term frequencies in the documents, we add new term(s) to FRT. We control the expansion of the FRT by several threshold parameters. The training algorithm is executed in an iterative way, and it ends when the performance cannot be further improved significantly. The detailed training algorithm is described below.

As the alternative statistical method, we use the K-nearest neighbour (KNN) algorithm that is one of the simplest classificatory algorithms, but shows good performance. In this actual version a document, having endowed K neighbours from the document collection, is assigned to category c, if at least θ out of its K neighbours are from the given category c ($\theta \in \boldsymbol{N}$, $\theta \le K$).

2 Preliminaries

Let us denote the given set of documents by D. Let W be the universal dictionary compiled from documents D containing all the significant words of the collection. In general, let C be the fixed finite set of categories. Each document $d \in D$ is classified into a subset of D. The classification can be considered as partition of the subject domain of documents. If one document is assigned to a category uniquely (no multiple category names are allowed) then the *partition* is *crisp*, otherwise *fuzzy*. For simplicity, let us first consider the crisp case, i.e. when documents are classified in nonempty pairwise disjoint *blocks*, each labelled with the corresponding category name. If a category name is too general for the classification purpose, we further refine the corresponding block by dividing it into more than one sub-block together with their category names newly labelled. Continue this process until a required refinement is reached. By means of the described technique multiple level categorization of documents is obtained, where categories are nested. Each category $c \in C$ has a certain level or deepness in the categorization, which is defined recursively by the function $n_C : C \to \boldsymbol{N}$ as

$$n_C(c) = \begin{cases} 0, \text{ if } c \text{ is a top level category} \\ n_C(c') + 1, \text{ otherwise} \end{cases} \qquad (1.1)$$

where c is directly obtained by the partition of category c'. The depth of a category set is defined as the level of the deepest category: $\text{depth}(C) = \max_{c \in C} n_C(c)$.

If we consider fuzzy partition of the subject domain, dividing the original partition could result in overlapped blocks. For example, consider Figure 1 where the category ``Cassette MP3 Player’’' belongs to both of the categories ``MP3 Player’’ and ``Cassette’’. Such a situation does not change the set of categories. Instead, the definition of category level, equation need to be modified slightly, because a category may have more than one parent category. We hence redefine (1.1) as

$$n_C(c) = \begin{cases} 0, \text{ if } c \text{ is a top level category} \\ \min_{c' \text{ is parent of } c}(n_C(c')+1), \text{ otherwise} \end{cases}$$

where c is directly obtained by the partition of category c'. Due to the construction of refinement, the top-level categories should be always pairwise disjoint.

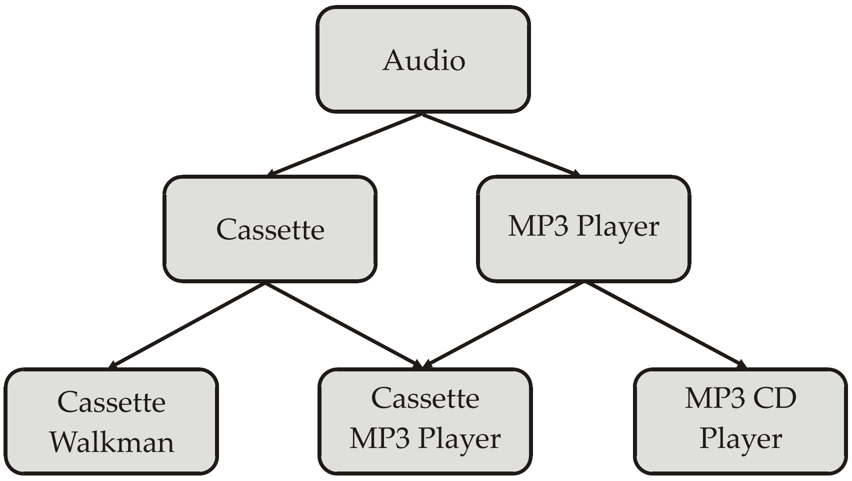

Figure 1: Multiple parentcraft

The refined category set, having a tree-like structure, offers a way to connect the categorization purpose and fuzzy relational thesauri [5]. The used FRT is described in details in [7], where each FRT term is either concept or instance. For the categorization purpose there is no difference between the two. We hence call FRT terms as concepts for simplicity. Let us denote the set of FRT terms by *T*. We can apply FRT based text categorization if the following condition holds: there must exist an onto mapping $f: T \to C$ satisfying $\forall c \in C : !\exists t \in T : f(t) = c$. The value of $f(t)$ is $\varnothing$ for such terms *t*, which do not take a part directly in the categorization. Because *f* is an onto mapping, its inverse f^{-1} exists. As a special case, category names can be FRT names as well: $C \subset T$. Usually $|C| \ll |T|$.

Analogously to (1.1) and , we can introduce the definition of level or deepness in the FRT terminology: $n_T : T \to \boldsymbol{N}$ that determines the level of certain FRT term in the case of crisp partition

$$n_T(t) = \begin{cases} 0, \text{ if } t \text{ is a top level concept} \\ n_T(t') + 1, \text{ otherwise} \end{cases} \qquad (1.3)$$

and in the case of fuzzy partition

$$n_T(t) = \begin{cases} 0, \text{ if } t \text{ is a top level concept} \\ \min_{t' \text{ is parent of } t} (n_T(t') + 1), \text{ otherwise} \end{cases}$$

where t' is the parent of concept *t*. For terms *t* having nonempty $f(t)$ values: $n_T(t) = n_C(f(t))$. If not stated otherwise, we will use the terminologies concept and category as synonyms, because the mapping *f* fixes their relation.

For FRT based text categorization we also need to process the FRT terms appearing in the documents. It means that three data about each document in the collection should be stored and maintained: words and FRT terms appearing in the document, and the categories into which it is classified. For modelling, we use the most common representation framework, the *vector space* model. Three vectors represent a document *d*: $\mathbf{W}_d$ describes the words, $\mathbf{C}_d$ the categories, and $\mathbf{T}_d$ the FRT terms of *d*. For the sake of better readability, the next notation uses (object, value) pairs or (object, value, counter) triples as vector elements, but when implementing the objects can be omitted:

$$\mathbf{W}_d = \left(\left\langle W_1, w_1^d, n_{w_1}^d \right\rangle, \ldots, \left\langle W_{|W|}, w_{|W|}^d, n_{w_{|W|}}^d \right\rangle, \; W_i \in W, w_i^d \in [0,1], n_{w_i} \in \boldsymbol{N} \right)$$
$$\mathbf{C}_d = \left(\left\langle C_1, c_1^d \right\rangle, \ldots, \left\langle C_{|C|}, c_{|C|}^d \right\rangle, \; C_i \in C, c_i^d \in [0,1] \right)$$
$$\mathbf{T}_d = \left(\left\langle T_1, t_1^d \right\rangle, \ldots, \left\langle T_{|T|}, t_{|t|}^d \right\rangle, \; T_i \in T, t_i^d \in [0,1] \right)$$

(1.5)

W_i denotes the *i*th word of the global dictionary, the corresponding w_i^d value is the relevance of term W_i to the characterization of the document *d*, and $n_{w_i}^d$ the number of appearance of term W_i in document *d*. Analogously, T_i denotes the *i*th term of FRT term set, the corresponding t_i^d value indicates the relevance of the term T_i to the characterization of the document *d*. Finally, C_i is the *i*th category, and c_i^d its weight for document *d*. If it is not ambiguous, the upper *d* indices of frequency and counter values can be omitted. For simplicity we also use the following sets:

$$W_d = \left\{ \left\langle W_i, w_i, n_{w_i} \right\rangle \mid w_i \geq 0 \right\} \text{ for words appearing in } d$$

$$T_d = \left\{ \left\langle T_i, t_i \right\rangle \mid t_i \geq 0 \right\} \text{ for FRT terms appearing in } d$$

$$C_d = \left\{ \left\langle C_i, c_i \right\rangle \mid c_i \geq 0 \right\} \text{ categories to which } d \text{ is assigned}$$

Note that a counter value n_{w_i} is nonzero if and only if the corresponding relevance value, w_i, is positive. There are numerous possible weighting schemes in the literature to determine the values of weights w_i, t_i and c_i. For word weights we used the most popular is the tf×idf weighting [8], which defines w_i in proportion to the number of occurrence of the term W_i in the document, f_i, and in inverse proportion to the number of documents in the collection for which the term occurs at least once: $w_i = f_i \cdot \log\left(\frac{N}{n_i}\right)$, where n_i the number of documents the word W_i occurs. The word vectors in (1.5) are normalized before any further processing is done. The FRT term weights t_i can be binary values, counters, or calculated by a more complicated weight measure, but in our experience counters are the most suitable choice. We remark that the significance of the selection of weights t_i depends on the weighting method we use for LUBS and CL based inference methods (see section *Weighting methods*). Due to the fact that since FRT may contain expressions consisting of more than one words, the word based model is impossible to apply. Therefore when implementing, it requires different technique from word procession. For c_i *category frequency* is an appropriate choice, which is usually binary. A document can have multiple categories or no category.

3 Categorization step

3.1 Category inference method

If we intend to infer the category of a document $d \in D$ by means of the FRT, the document should contain words from the FRT, i.e. the following condition should hold: $|T_d| = k \geq p_0$, $p_0 \in \boldsymbol{N}$ parameter determines the minimum number of FRT terms to apply FRT categorization (usually 1). We refer to set consisting the first member of the pairs as $T_d^* = T_d|_T$. We call elements of T_d^* as *supporting FRT-terms* w.r.t. to a category *c*, if they take part in the determination of *c*, being the result of categorization. When this condition is satisfied, we can use the FRT to determine the category/ies to which the given document belongs. Based on the set T_d, we can infer the category/ies of document *d* in various ways, moreover, these inference methods can be combined. Due to the lack of space we present here only the best inference method.

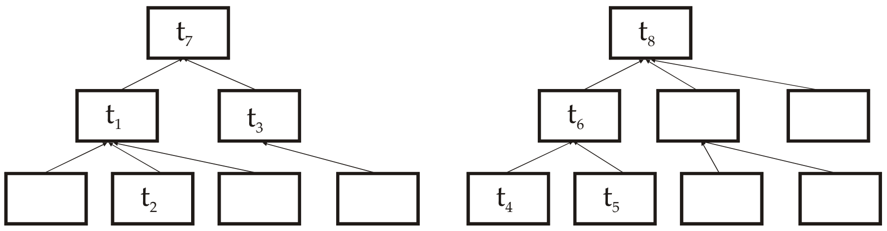

Figure 2: Example for categorization. FRT-terms of the current document are: $T_d = \{\langle t_1, 13\rangle, \langle t_2, 9\rangle, \langle t_3, 1\rangle, \langle t_4, 1\rangle, \langle t_5, 1\rangle\}$***. Meaning of notation:*** t_1 ***– Printer;*** t_2 ***– Laser Printer;*** t_3 ***– Monitor;*** t_4 ***– Graphic Card;*** t_5 ***Sound Card;*** t_6 ***– Multimedia Kit;*** t_7 ***– Computer Peripheral Devices;*** t_8 ***– Computer Components***

Before we turn to the discussion of the inference method in details, we recall a function from [7] operating on FRT terms. Function $\text{Sup}: T \to 2^T$ calculates the set of the operand's all super concepts (see [7, p.19]):

$$\text{Sup}(t^*) = \{t \in T \mid t^* \leq_T t\} \qquad (1.6)$$

where $t_1 \leq_T t_2$ means that t_2 is more general term (instance or concept) than t_1 for $t_1, t_2 \in T$. By definition, $t \leq_T t$ holds for all $t \in T$. Here, 2^A denotes the power set of the set *A*. For brevity, $\text{Sup}(t^*)$ is called the *FRT-superset* of t^*. Analogously, we can define *FRT-subset* of a term [7].

Let us now turn to the discussion of the best method for category determination, the Concept Level (CL) based method. This method determines the categories of a document at each concept level separately. For each $t_i \in T_d^*$, we create its FRT-superset by means of (1.6). As an example consider the following case depicted in Figure 2. The actual document *d* contains FRT-terms T_d^*:

``Printer'', ``Laser Printer'', ``Monitor'' under top-level concept ``Computer Peripheral Devices'' (CPD), moreover ``Graphic Card'' and ``Sound Card'' under top-level concept ``Computer Components'' (CC). Then $\mathrm{Sup}(t_1) = \{t_1, t_7\}$, $\mathrm{Sup}(t_2) = \{t_1, t_2, t_7\}$, $\mathrm{Sup}(t_4) = \{t_4, t_6, t_8\}$, etc. We applied different weighting methods to rank the concepts of $\mathrm{Sup}(t_i)$; these are presented in section *Weighting methods*. As the result of weighting, we obtain set of (term, value) pairs for each $t_i \in T_d^*$. Among them we select the final term as

$$t_{\mathrm{CL}}^{(i)}(t_i) = \left\{ \left\langle t, w_i^t \right\rangle \mid t \in T, w_i^t \text{ determined by weighting} \right\}$$

For the final ranking w_i^t weights are cumulated, for every *t* and for every $i \in \left[1, \left|T_d^*\right|\right]$, i.e. if a term *t* appears in more than one $\mathrm{Sup}(t_i)$ set, then in the overall ranking these weights of *t* are added:

$$t_{\mathrm{CL}}^{(i)}(t_d^*) = \left\{ \left\langle t, \sum_{i=1}^{\left|T_d^*\right|} w_i^t \right\rangle \middle| \exists i \in \left[1, \left|T_d^*\right|\right] : t \in \mathrm{Sup}(t_i) \right\} \quad (1.7)$$

where $w_i^t = 0$ by definition if $t \notin \mathrm{Sup}(t_i)$. When determining the category at a certain concept level, the concept that appears at the given level with highest rank in $t_{\mathrm{CL}}^{(i)}(t_d^*)$ is selected. If there is more than one concept with the highest rank, then all of them are selected. This selection method is similar to the voting classification used in [2], but instead of decision trees, we evaluate FRT-supersets of FRT terms appearing in the documents. The final categories of *d* are the union of categories assigned to the selected concepts from $t_{\mathrm{CL}}^{(i)}(t_d^*)$ for each concept level.

If $\left|T_d\right| < p_0$ then the FRT is not suitable for determining the category/ies of *d*, especially when $p_0 = 1$. In this case, we skip the categorization of the given document at the start. After the FRT is expanded, the cardinality of $\left|T_d\right|$ is increased considerably. The expansion algorithm guarantees this feature, if the parameters are set reasonably.

3.2 Weighting methods

In order to rank the resulted concepts that were determined by CL based method we employ weighting schemes. For every resulted concept *t*, the strength of relationship, $r(t,t_i)$, can also be retrieved between terms $t_i \in T_d^*$ and *t*. It is defined as

$$r(t,t_i) = \begin{cases} 1, \text{ if } t = t_i \\ r(t',t_i) \cdot r(t,t'), \text{ otherwise} \end{cases} \qquad (1.8)$$

where t' is the ``next'' term of the FRT hierarchy towards *t*. The weighting schemes assign a weight to each retrieved concept by means of $r(t,t_i)$ and t_i^d.

As we mentioned before, CL based method determines the category of a document at each category level separately. Without the loss of generality, let us now consider level $l \leq \text{depth}(C)$, and let us assume that we have N_l categories at level *l*. We intend to determine to which *l*-level category the actual document *d* belongs. The CL based method checks each $\text{Sup}(t_i)$ set for *l*-level concepts, and endows them a weight, so we obtain

$$t_{\text{CL}}^{(i)}(t_i) = \left\{ \left\langle t, w_i^t \right\rangle \mid n_T(t) = l \text{ and } w_i^t = \chi_{\text{Sup}(t_i)}(t) \right\} \qquad (1.9)$$

where χ is the usual characteristic function. We can use other, more sophisticated weighting instead of χ, which takes into account the number of appearance of a term, and/or the strength of relations connecting the element of $\text{Sup}(t_i)$ and the FRT term t_i. Reasonable alternatives are:

$$w_i^t = w_{\text{Sup}(t_i)}(t) = \begin{cases} t_i, \text{ if } t \in \text{Sup}(t_i) \\ 0, \text{ otherwise} \end{cases} \qquad (1.10)$$

$$\text{or } w_i^t = w_{\text{Sup}(t_i)}(t) = \begin{cases} t_i \cdot r(t,t_i), \text{ if } t \in \text{Sup}(t_i) \\ 0, \text{ otherwise} \end{cases}. \qquad (1.11)$$

Weighting scheme with expression (1.10) gives higher weights to concepts with multiple appearances of the supporting FRT terms, but the strength of relation between the selected concept and the FRT term is not taken into account. The usage of (1.11) eliminates this deficiency, incorporating the latter information.

Let us investigate the example of Figure 2 for the analysis of weighting schemes for CL method when $l = 0$ (top-level). $t_7 \in \text{Sup}(t_1), \text{Sup}(t_2), \text{Sup}(t_3)$ and $t_8 \in \text{Sup}(t_4), \text{Sup}(t_5)$. We calculate the aggregated weights for t_7 and t_8 based on (1.9) using the above three weighting schemes: $w_{t_7} = 3$, 23, and 19.89 according to (1.9), (1.10) and (1.11), resp.; analogously $w_{t_8} = 2$, 2, and 1.62. Regardless of the employed weighting scheme, we select t_7 as top-level concept.

4 Training FRT: expansion step

In order to improve the categorization capability of FRT, we may need to check the correctness of the selected categories and modify or insert new terms into the FRT. Let C_d^{kNN} denote the set of determined categories by KNN method for document *d*, and $N_d^{(k)}$ the *k* nearest neighbour of $d \in D$. As preprocession for FRT expansion, we create for each category $c \in C$ in C_d an *expansion set* E_c^d. This can be considered as a small local dictionary that captures the key words for given a document and category. E_c^d is an ordered set of important words appearing in *d* and in those of its neighbouring *k* documents, which belongs to category *c*. E_c^d is truncated to keep only the first p_1 words having at least α_1 weight. (In accordance with [5], this set is the (p_1, α_1)-level set of E_c^d.) In the following we add terms to the FRT based on E_c^d expansion sets.

For FRT expansion we check for each document $d \in D$ whether its original categories were found by FRT categorization, and whether it was assigned to categories incorrectly.

1) Case: $c \in C_d$ but $c \notin C_d^{\mathrm{FRT}}$ (the FRT was unable to find the correct category *c*). Let us consider each element *w* of set E_c^d, and let $t = f^{-1}(c)$, the FRT term representing the category *c*. Check whether the term *w* has been added in the expansion process to the FRT before (note: same word(s) can appear in different E_c^d sets). If not (*insertion of a new term*): link word *w* as a new term to the FRT under the term *t* with the predefined relation weight. Figure 3a depicts this situation. If the term *w* has been already added to the FRT during the expansion process, while it was linked to one of the element $t_{\mathrm{old}} \in \mathrm{Sup}(t)$ (*correction of a previously added relation*): delete the relation between the higher-level concept t_{old} and the term *w*, and add a new relation with the predefined relation weight to the FRT between $t = f^{-1}(c)$ and *w*. This operation is

depicted on Figure 3b. Moreover, if term *w* has been already added to the FRT, and it is linked under a concept t' that is neither in the FRT-superset (located above) nor in the FRT-subset (located under) of *t* (*insertion of a new relation to an existing term*): add a new relation with the predefined relation weight to the FRT between *t* and *w*. Observe on Figure 3c that the level of the parents of term *w* is not necessarily the same.

2) Case: $c \notin C_d$ but $c \in C_d^{\mathrm{FRT}}$ (the category determined by the FRT is incorrect). Modify the weights of relationships between the incorrect *c* concept (category name) and those FRT terms $t \in T$, which supported the selection of the incorrectly determined category. If $r\left(f^{-1}(c),t\right)$ is such a relation then multiply its weight by the factor α_2 ($\alpha_2 \in (0,1) \subset \boldsymbol{R}$ adjustable parameter).

3) Otherwise do nothing.

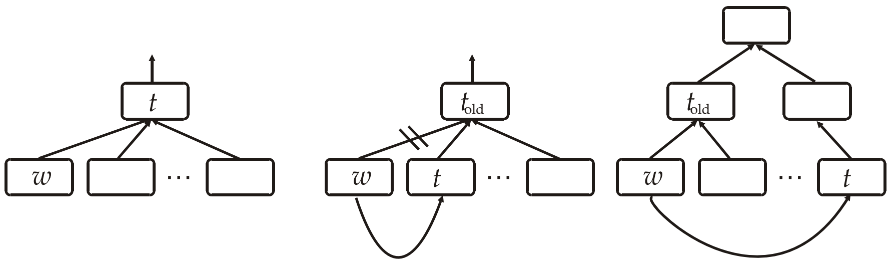

Figure 3: Insertion of a new term

Recalculate the FRT-supersets based on the new terms and modified weights, and repeat step 1–3 until the terminal condition of the training iteration is not reached.

5 Implementation and experimental results

5.1 Document collection

Because there is no standard document corpus particularly for multi-class and multi-label text classification test, we collected documents in the domain of *electronic appliances* from the web. First we collected 211 documents, which was augmented later, thus a 327 elements document collection was obtained. We proceeded tests on both document sets. The dictionary consisted of 3731, and 5793 words, respectively. FRT was created by semi-automatic thesaurus construction software [7], which allows fuzzy partition of subject domain. In our application we use three embedded category levels, we term them topic, subtopic, and subsubtopic, respectively.

The collected documents were classified into the following six top concepts: Audio, Computer, Computer Components, Computer Peripheral Device, House-Hold Appliances, Office Communications Appliances. We had 30 (31) subtopics, and 40 (58) subsubtopics in the case of 211 (327) document set. Each category had at least two training examples. The training documents were distributed evenly, except the Computer Components topic that had 62 (178) documents. The document collections, and the mapping files containing the assignment between category names and FRT terms are available publicly at `http://ozzy.chonbuk.ac.kr` under *FRTcat* section.

5.2 Result of the FRT categorization

We proceeded experiences with CL based inference methods to determine categories. We proceeded in three steps. First the top-level category of a given document was determined. In consecutive steps we searched the actual subtopic or subsubtopic only under the already determined topic or subtopic, knowing that categories were embedded. This directed search resulted in significantly better performance than undirected search, which considered all the categories.

We tried all the weighting methods presented earlier in the paper. The most sophisticated method (1.11) gave the best result. We fixed $\alpha_2 = 0.8$.

We intend to add only some remarks to the easily interpretable results shown in Table 1. The speed of training is considerable faster in the first 3–4 training iterations, usually at this stage the performance differs from the best result by only 2–3 percentage. Obviously, the number of inserted terms is much higher at this stage of training: in the first cycle more than 500 terms can be added e.g. in the case of 327-element document set, while later only a few tens of terms are inserted. In general, the performance of FRT categorization is not monotone increasing in terms of the number of iteration, it can oscillate under the optimal value. We remark that the optimal number of neighbours for expansion set creation is 8, although KNN gives better results with smaller values.

Table 1: Results with the 211-elements and 327-elements category set (separated with double line)

k	F-measure	Precision	Recall	p_1	α_1	Number of iterations

8	0.9642	0.9818	0.9474	5	0.10	8
4	0.9587	0.9799	0.9385	5	0.10	10
8	0.9723	0.9927	0.9562	7	0.05	11
4	0.9568	0.9815	0.9333	7	0.05	12
4	0.9473	0.9823	0.9146	10	0.10	9
8	0.9434	0.9777	0.9114	10	0.10	9
8	0.9560	0.9860	0.9278	7	0.05	12

6 Conclusion

We proposed a new method for text categorization, which uses FRT to support the classification task. The algorithm is particularly efficient if category set have multilevel structure. In our experiments we achieved good results, around 95–97 percentage, for F-measure value. The method exploits the adaptive feature of the local dictionaries stored in FRT. We intend to apply our method to the Reuter-21578 benchmark data collection for comparison with conventional methods, and to make experiments with different FRT expansion algorithms in the near future.

References

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

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

[3] Y. Yang: An evaluation of statistical approaches to text categorization. *Information Retrieval*, **1**(1–2), pp. 69–90, 1999.

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

[5] H.L. Larsen, 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), pp. 31–40, 1993.

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

[7] J.H. Choi, J.J. Park, J.D. Yang, 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