Experiments with HITEC: a Hierarchical Text Categorizer*

Domonkos Tikk[1] and György Biró[2]

[1] Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
H-1117 Budapest, Magyar Tudósok körútja 2., Hungary
Email: tikk@tmit.bme.hu

[2] Department of Informatics, Eötvös Loránd Science University
H-1117 Budapest, Pázmány sétány 1/c, Hungary
Email: gbiro@tmit.bme.hu

Abstract

This paper presents experiments on the effectiveness of HITEC software (HIerarchical TExt Categorizer) on several natural languages (English, German) and with various kinds of text corpora. HITEC applies UFEX (Universal Feature EXtractor) method for hierarchical text categorization. Based on the obtained results shows that HITEC outperforms its known competitors on the investigated corpora, its performance is independent from the processed languages. The time and storage requirement of HITEC is considerable, therefore it can be run on an average PC.

1 Introduction

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. A text classifier assign a document to appropriate category/ies, also called *topic*, in a predefined set of categories.

Originally, research in text categorization (TC) 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, 15, 17]. 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 [11], multi-label problems have received very little attention [1].

*This work was funded by Grant FKFP 0180/2001.

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 databases are often organized as hierarchies; see e.g. Yahoo and WIPO's patent database[2].

Patent databases are typically such where the use of a hierarchical category system is a necessity. Patents cover a very wide area of topics, and each field can be further divided into subtopics, until a reasonable level of specialization is reached. (The hierarchy of topics are often called *taxonomy*.) The International Patent Classification (IPC) is a standard taxonomy developed and administered by WIPO (World Intellectual Property Organization) for classifying patents and patent applications. As a courtesy of WIPO, we could experiment with the WIPO-alpha English and WIPO-de German patent databases issued in late 2002 and early 2003 respectively[3]. WIPO-alpha is a large collection (3 GB) of about 75000 XML documents distributed over 5000 categories in four levels (the top four level of IPC); WIPO-de is an even larger collection of about 110000 XML documents defined on the same taxonomy (IPC). Due to the strongly business sensitive nature of the research, studies on WIPO collections are not publicly available. Our primary purpose with this database is to analyze the applicability of our algorithm on a very large real-world collection in terms of efficiency and feasibility (time and space requirements).

Beside the WIPO-alpha collection, we tested our software on two other document corpora, being the well-known Reuters-21578 document collection.

The paper is organized as follows. Section 2 give an overview on UFEX and the major features implemented in HITEC. Section 3 report on our experience. The conclusion is drawn in Section 5.

2 Categorization based on UFEX

UFEX (Universal Feature Extractor) method aims at determining relevant characteristics of a set of categories based on training entities. It is particularly optimized to handle hierarchically organized category structures. The nature of the training entities is independent from UFEX as it applies an internal representation form, therefore it is able to work on arbitrary kind of data (e.g. text, image, numerical measurements) that can is described by a numerical vector of features. The basic idea of UFEX is described in details in [12, 13]. For simplicity, in the next brief overview and later on we will use the TC-specific notations. Here we remark again that, nevertheless, UFEX is designed to be able to process arbitrary numerical data.

The core idea of UFEX 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 and assign the document to the category of the most similar descriptor. When this procedure fails finding correct category we raise the weight of such features in the category descriptors that appear also in the given document. If a document is assigned to a category incorrectly, we lower the weight of such features in the descriptors that appear in the document. We tune category descriptors by finding the optimal weights for each

[1] In general hierarchy is considered to be an acyclic digraph; in this paper we restrict our investigation to tree structured hierarchies.

[2] `http://www.yahoo.com`, `http://www.wipo.int/ibis/`

[3] The collections are available after registration at `http://www.wipo.int/ibis/datasets/index.html`

feature 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 2.2 about the training algorithm. For test documents the classifier works in one pass by omitting the feedback cycle.

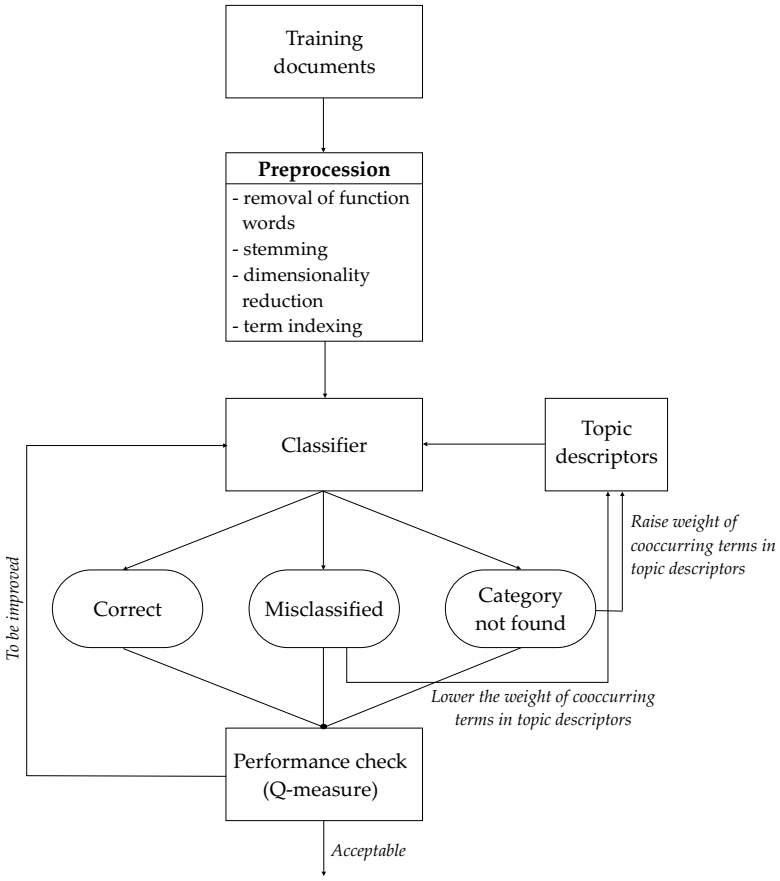

Figure 1: The flowchart of the training algorithm of UFEX

The rest of this section is organized as follows. Subsection 2.1 describes the topic hierarchy, vector space model and descriptors. Subsection 2.2 presents classification and the training method.

2.1 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.

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, $d \in \mathcal{D}_{\mathrm{Train}}$, and test documents, $d \in \mathcal{D}_{\mathrm{Test}}$, where $\mathcal{D}_{\mathrm{Train}} \cap \mathcal{D}_{\mathrm{Test}} = \emptyset$, and

$\mathcal{D}_{\text{Train}} \cup \mathcal{D}_{\text{Test}} = \mathcal{D}$. 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 document d_j belongs to along the topic path (c_1 is the highest). Note that the root is not administrated in the topic set, as it owns all documents. As a consequence of our previous condition, c_q is a leaf-category. 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, 6], 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*

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

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 experimented with the well-known tf$\times$idf and entropy weighting, and found the latter being much more effective, although slower to determine.

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

$$\text{descr}(c_i) = \langle v_{1i}, \ldots, v_{|\mathcal{T}|i} \rangle, \quad c_i \in \mathcal{C} \tag{3}$$

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

2.2 Classification and training

2.2.1 Classification

When classifying a document $d \in \mathcal{D}$ its term vector (2) is compared to topic descriptors (3) and based on the result the classifier selects (normally) a unique category. At a given stage of the classification only a subset of topic are considered as potential category; this is based on the following greedy selection process. 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 at an arbitrary stage of the classification of document d_j we have to select from k categories: $c_1, \ldots, c_k \in \mathcal{C}$. 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:

$$\mathrm{conf}(d_j, \mathrm{descr}(c_i)) = f\left(\sum_{k=1}^{|\mathcal{T}|} w_{kj} \cdot v_{ki}\right), \tag{4}$$

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 [10] 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 $\mathrm{conf}_{\min} \in [0, 1]$, i.e. the greedy selection algorithm continues when

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

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

2.2.2 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 (a document is classified incorrectly), we modify term weights in category descriptors.

The classifier can commit two kinds of error: 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 the decisions 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

$$\delta(v_{ki}) = \alpha \cdot (\mathrm{conf}_{\mathrm{req}} - \mathrm{conf}(d_j, \mathrm{descr}(c_i)))) \cdot w_{ij}, \quad 1 \leq k \leq \mathcal{T} \tag{5}$$

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

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 $1 \leq k \leq \mathcal{T}$. 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

$$\begin{aligned} \delta^{(n+1)}(v_{ki}) = {} & \alpha \cdot (\mathrm{conf}_{\mathrm{req}} - \mathrm{conf}(d_j, \mathrm{descr}(c_i)))) \cdot w_{ij} \\ & + \delta^{(n)}(v_{ki}) \cdot \beta \end{aligned} \tag{6}$$

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 reaches a quasi-maximal value. We use average of Q-measure (defined in [12]), $\overline{Q}$, for inter-training performance measure, because it is more sensible to small changes in the effectiveness of the classifier than, e.g., F-measure [14]. By setting a maximum variance value $\max_{\text{var}}$ (typically $0.9..1.0$) we stop training when actual $\overline{Q}$ drops below the $\max_{\text{var}} \cdot \overline{Q}^{\text{best}}$, where $\overline{Q}^{\text{best}}$ is the best $\overline{Q}$ achieved so far during training.

3 Experimental results

3.1 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 $|\mathcal{T}'| \ll |\mathcal{T}|$ [11]. 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. [9, 16]). In our previous experiments [13] 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 $|\mathcal{T}|$ by disregarding terms that either occur less than $\min_{\text{occur}}$ times in the entire train document set, or occur more often than a certain threshold in the training set, i.e. if $n_k/|\mathcal{D}_{\text{Train}}| \geq \max_{\text{freq}}$. By the former process we disregard words that are not significant in the classification, while by the later process we ignore words that are not discriminative enough between categories. The typical values are $\min_{\text{occur}} \in [1 \,..\, 10]$ and $\max_{\text{freq}} \in [0.05 \,..\, 1.0]$.

3.2 Performance measures

Beside the usual recall, precision, and F-measure, we have adopted three heuristic evaluation measures that were proposed in [7]. Let us suppose that algorithm returns an ordered list of guesses (IPC codes in case of WIPO collections), where the order is determined by the confidence level (see (4)) used as weight. Then we can define the following measures (see Figure 2):

1. **Top prediction** (briefly: Top) The top category predicted by the classifier is compared with the main IPC class, shown as [mc] in Figure 2.

2. **Three guesses** (Top 3) The top three categories predicted by the classifier are compared with the main IPC class. If a single match is found, the categorization is deemed successful. This measure is adapted to evaluating categorization assistance, where a user ultimately

makes the decision. In this case, it is tolerable that the correct guess appears second or third in the list of suggestions.

3. **All classes** (Any) We compare the top prediction of the classifier with all classes associated with the document, in the main IPC symbol and in additional IPC symbols, shown as (ic) in Figure 2. If a single match is found, the categorization is deemed successful.

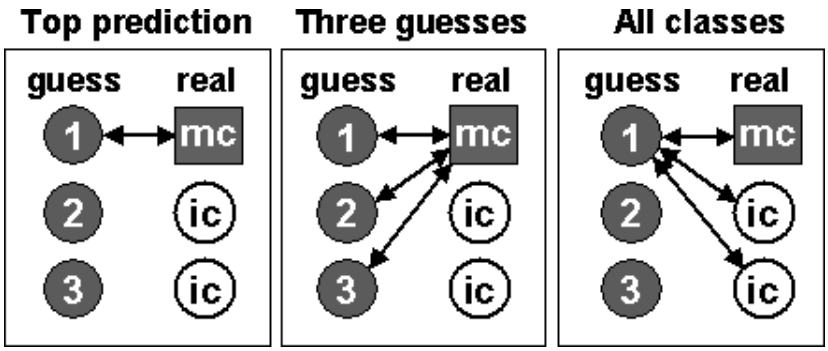

Figure 2: Explanation to the three evaluation measures Top, Top 3, Any [7]

Although in [7] it is suggested to use these measures solely on IPC class level, we have applied them on each level of the taxonomy.

3.3 Document collections

This document collection[4] has been the most widely used benchmark corpus among researchers on the field of text categorization. A comparative study of different approaches using flat, i.e. not-hierarchical category system, on this corpus can be found in [11, page 38]. 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 original categories into hierarchies. One of the first works on hierarchical text classifiers, Koller and Sahami, [9], also reported on experiences with two taxonomies on the subset of Reuters collection. Chakrabarti et al [4] also applied their hierarchical text classifier on this corpus but without taxonomy. D'Alessio et al defined 3 different hierarchies [5, page 10–11] (Experiments E3, E4, E5) that comprise all categories of the original flat settings and they achieved better numbers in accuracy that has been reached so far by flat categorizers. We applied D'Alessio's and the flat taxonomies on the Reuters corpus to compare our algorithm to the others, because we these were the only explicitly given ones that we could reproduce. We have adopted also 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

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

thousands training documents, and with no documents at all. The number of categories without training and test documents is 19.

Table 1 presents the results achieved with the E3–E5 taxonomies [5] and with the regular flat category system (we indicate the reference results of D'Alessio et on the right hand side of Table 1). A quasi-optimal setting of parameters $\mathrm{conf}_{\min}$, $\mathrm{max}_{\mathrm{var}}$ is also indicated; one may observe that there is no significant difference in their values at the various taxonomies. We remark that our results shows the same relationship among taxonomies as reported by D'Alessio et al: E4 yields the best result. Wibowo and Williams [16] experienced this collection with another hierarchy [8], 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. The obtained result, 89.43%, is somewhat better than the best known results of flat categorizers (87.8% Weiss et al [15], committee of decision trees).

Table 1: Results with E3–E5 and flat taxonomies on Reuters-21578 collection

	Our method			D'Alessio et al		
taxonomy	F_1	π	ρ	F_1	π	ρ
E3	90.7	90.2	91.3	79.8	81.4	78.3
E4	92.8	92.5	93.1	82.8	85.9	79.9
E5	92.5	92.0	92.9	82.5	86.4	79.0
flat	88.6	88.5	88.7	78.9	80.3	77.6

3.4 The WIPO-alpha collection

The WIPO-alpha collection consists of 3 GB XML documents (in total about 75000 documents); documents are assigned one main category, and can be also linked to several other categories. The IPC taxonomy has four levels termed: section, class, subclass, and main group (top-down). The collection is provided as two sub-collection of a training set of 46324 documents, and a test set of 28926 documents. The training collection consists of documents roughly evenly spread across the IPC main groups, subject to the restriction that each subclass contains between 20 and 2000 documents. The test collection consists of documents distributed roughly according to the frequency of a typical year's patent applications, subject to the restriction that each subclass contains between 10 and 1000 documents. All documents in the test collection also have attributed IPC symbols, so there is no blind data.

Each document includes a title, a list of inventors, a list of applicant companies or individuals, an abstract, a claims section, and a long description. IPC codes (topic labels) have been attributed to each document. A very detailed description of the collection can be found in [7].

We present the obtained results by means two figures (Figure 3–4) and a summarizing table (Table 2). We differentiated results based on the confidence level. Here 0.0 means that all guesses are considered, while 0.8 means that only those decision are considered where the confidence level is not less than 0.8. Obviously, the higher is the confidence level, the lower is the number of considered documents. The figures show all the three performance measures at class, subclass and main group levels by increasing confidence levels of 0.1 step. Table 2 compares some significant values of each parameter setting.

The best results have been achieved after 7 iterations when only the fields of *i*nventors,

*a*pplicants, *t*itle, *c*laims and *a*bstract are used for dictionary creation ("iptca" setting); entropy weighting is used; $\min_{\text{occur}} = 2$ and $\max_{\text{freq}} = 0.25$. (Figures 3). The other settings, e.g. "ipta" that appears in Table 2, are modification of this one.

We investigated the effect if only the main category of each patent document is used for training. This experiment was suggested by the developers of the collection [7], and can be argued that this selection makes ambiguous training documents (having more topics) unique for training purpose. The obtained result did not support this hypothesis, the obtained results were inferior than the ones with regular setting (Table 2).

The next experiments were obtained when semantic information were propagated back to the learning phase. This modification takes into account the location of the clue word in a sentence and the location of an important sentence in a paragraph. Cumulating these information we can determine areas in the text that are more important than others. This modification has great effect on the results at low confidence levels since it increases certain performance measure values by more than 3%. See figure 4 and the Table 2.

It worth to note that the graph of Top 3 measure is dropping when the consistency level increases. The reason is that at lower consistency level more categories are returned, and based

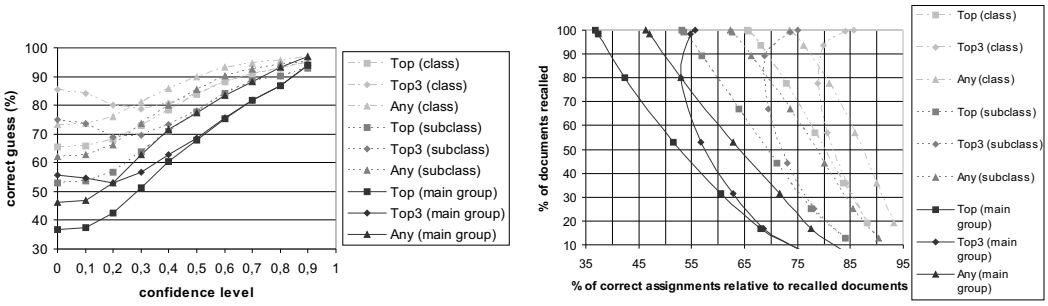

Figure 3: Setting "iptac". Precision by confidence levels. b) Comparisons of precisions, extrapolated to 100% recall

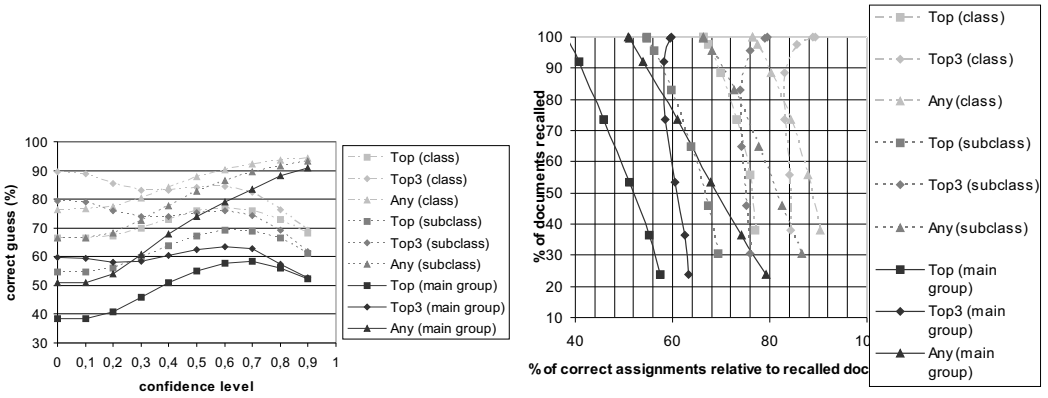

Figure 4: Use semantic information. a) Precision by confidence levels. b) Comparisons of precisions, extrapolated to 100% recall

Table 2: Summary of results on WIPO-alpha patent collection

Eval. ms.	Set	IPC/conf. level			
		cl./0.0	cl./0.8	s.cl./0.0	m.g./0.0
Top	ipta	65.75	92.93	53.25	36.89
	iptac	65.50	92.93	53.14	36.78
	main	62.81	84.37	49.41	32.28
	sem.	66.41	72.86	54.63	38.38
Top3	ipta	85.56	92.93	75.05	55.44
	iptac	85.61	92.93	75.00	55.58
	main	83.61	84.37	70.89	48.98
	sem.	89.41	76.45	79.48	59.64
Any	ipta	73.68	95.83	62.45	46.46
	iptac	73.41	95.64	62.28	46.38
	main	71.32	94.97	58.97	41.51
	sem.	76.46	93.48	66.36	50.90

on the results, in some case the one returned with lower consistency can be the correct one, but that is left out at higher consistency levels. When the consistency level goes higher, much fewer documents are considered, and then the Top 3 values increases again.

One can observe that the relationship between evaluation measures is Top $<$ Any $<$ Top 3 except when tf$\times$idf weighting scheme is applied: then Any gives the best values. Naturally, the lower we go in the taxonomy the more imprecise predictions are. At IPC classes the Top 3 measure attains 85.5% at several settings with the lowest confidence level (i.e. when basically all documents are considered) that is a quite significant result. This value hints that the algorithm can be used for large document corpora in real-world applications. Because of the very large taxonomy and range of documents the results on the main group level seems to be quite weak. However, if we consider that human experts can do this categorization with about 64% accuracy then this result turns out to be much more significant.

Let us shortly remark the time efficiency of the method. Our experiments were executed on a regular PC (Linux OS, 2 GHz processor, 1 GB RAM). The indexing of the entire train collection took around one hour with entropy weighting and just over 40 minutes with tf$\times$idf weighting. The training algorithm (7 iterations) required about 2 hours with each settings. If more iterations were done the results did not improved significantly.

4 The WIPO-de collection

The WIPO-de collection is very similar to WIPO-alpha, except that it contains German patent applications. The collection is provided as two sub-collection of a training set of 84822 documents, and a test set of 26006 documents. In Table 3 we present selected results with the best setting (using semantic information).

Let us shortly remark the time efficiency of the method. Our experiments were executed on a regular PC (Linux OS, 2 GHz processor, 1 GB RAM). The indexing of the entire train collection took around one hour (with entropy weighting requiring two passes for each document). The

Table 3: Summary of results on WIPO-de patent collection

Eval. ms.	IPC/conf. level			
	cl./0.0	cl./0.8	s.cl./0.0	m.g./0.0
Top	65.02	86.95	55.37	37.93
Top3	87.14	89.00	77.61	57.34
Any	75.04	96.95	66.88	50.79

training algorithm (7 iterations) required about 2 hours with each settings. If more iterations were done the results did not improved significantly.

5 Conclusion

In this paper we showed the effectiveness of HITEC that is based on the UFEX method on different document corpora. The time and storage requirement of the software allows to learn large document corpora in reasonable time even on a regular PC and to process unknown/test documents in real time.

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] 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`.

[6] 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.

[7] C. J. Fall, A. Törcsvári, and G. Karetka. Readme information for WIPO-alpha autocategorization training set, December 2002. `http://www.wipo.int/ibis/datasets/wipo-alpha-readme.html`.

[8] 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.

[9] 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.

[10] 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`.

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

[12] D. Tikk and Gy. Biró. Experiment with a hierarchical text categorization method on the WIPO patent collection. In *Proc. of the 4th International Symposium on Uncertainty Modeling and Analysis (ISUMA 2003)*, pages 104–109, University of Maryland, USA, September 21–24, 2003.

[13] D. Tikk, J. D. Yang, and S. L. Bang. Hierarchical text categorization using fuzzy relational thesaurus. *Kybernetika*, 39(5):583–600, 2003.

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

[15] 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.

[16] 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.

[17] 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.