Computer Science and Information Systems 14(1):103–121 DOI: 10.2298/CSIS151017030G

Hierarchical vs. flat n-gram-based text categorization: can we do better?

Jelena Graovac[1], Jovana Kovačević[1,2], and Gordana Pavlović-Lažetić[1]

[1] Faculty of Mathematics, University of Belgrade
Studentski trg 16, 11000 Belgrade, Serbia
{jgraovac,jovana,gordana}@matf.bg.ac.rs
[2] School of Informatics and Computing, Indiana University
Bloomington, Indiana, USA

Abstract. Hierarchical text categorization (HTC) refers to assigning a text document to one or more most suitable categories from a hierarchical category space. In this paper we present two HTC techniques based on kNN and SVM machine learning techniques for categorization process and byte n-gram based document representation. They are fully language independent and do not require any text preprocessing steps, or any prior information about document content or language. The effectiveness of the presented techniques and their language independence are demonstrated in experiments performed on five tree-structured benchmark category hierarchies that differ in many aspects: Reuters-Hier1, Reuters-Hier2, 15NGHier and 20NGHier in English and TanCorpHier in Chinese. The results obtained are compared with the corresponding flat categorization techniques applied to leaf level categories of the considered hierarchies. While kNN-based flat text categorization produced slightly better results than kNN-based HTC on the largest TanCorpHier and 20NGHier datasets, SVM-based HTC results do not considerably differ from the corresponding flat techniques, due to shallow hierarchies; still, they outperform both kNN-based flat and hierarchical categorization on all corpora except the smallest Reuters-Hier1 and Reuters-Hier2 datasets. Formal evaluation confirmed that the proposed techniques obtained state-of-the-art results.

Keywords: hierarchical text categorization, n-grams, kNN, SVM.

1. Introduction

Text categorization (TC) is the task of classifying unlabelled natural language documents into a predefined set of categories. In TC research, most of the studies have focused on flat text categorization (FTC) where the categories to be predicted are treated in isolation and there is no structure defining the relationships among them. However, organizing the categories into a hierarchy helps us solve the problem of browsing and searching the categories when their number grows significantly. Also, many important real-world categorization problems are naturally cast as hierarchical text categorization (HTC) problems, where the predefined categories are organized into a category hierarchy [34].

There are many HTC techniques that differ in a number of criteria. One of the criteria is the type of hierarchical structure used: *tree* (a node can have only one parent node) or *directed acyclic graph* (a node can have more than one parent node). The second criterion

is related to how deep the categorization in the hierarchy is performed: *mandatory leaf-node prediction* (always will classify a leaf node) or *non-mandatory leaf node prediction* (the classification can stop in any node at any level of the hierarchy). The third criterion is related to how the hierarchical structure is explored: *big-bang* classifiers, when a single classifier is used, coping with the entire category hierarchy; *top-down* classifiers, when the system employs a set of local classifiers; or *flat* classifiers, which ignore the category relationships, typically predicting only the leaf nodes [34].

Regardless of which HTC technique is chosen, we are faced with many different challenges and difficulties. One of the difficulties is high dimensionality and variable length, content and quality of text data. A huge number of text documents on the Web contains different kinds of textual errors, such as typing, spelling and grammatical errors. To complicate matters, much of that text is imperfect, having been derived from existing paper documents by means of an error-prone scanning and character recognition process. TC has to perform reliably on all inputs, and thus has to tolerate these kinds of problems to some extent. Although most of the research activity has concentrated on English text, the management and study of TC in languages other than English is a growing need. This introduces a number of additional difficulties in text analysis due to specific characteristics of different languages. For example, Asian languages such as Chinese are based on fundamentally different linguistic paradigm compared to Latin-alphabet languages. They use unbound morphemes and depend on post-phrase affixes and word order to convey meaning.

Standard approaches to TC [16] are usually based on traditional word-based vector document representation (e.g., bag-of-words, BOW). Although representation of a document at the level of words seems to be an intuitive solution, in some languages it could be a particular problem. For example, Chinese does not have word boundaries explicitly marked in text so word segmentation itself is a difficult problem. One of the main problems with the BOW technique is that word order information is lost. It ignores the fact that morphological features can also play an important role in text analysis. Traditional preprocessing of documents, such as eliminating stop words, pruning rare words, stemming and normalization, may improve the representation but at the same time produce a number of potential drawbacks: they may require a linguist (or a polyglot) for initial setup and subsequent tuning, they are vulnerable to variant spellings, misspellings and random character errors, and they tend to be both topic-specific and language-specific [6].

The main goal of this paper is to present two HTC techniques that are fully language independent so as to be efficiently applied to large amount of hierarchically organized text documents in different natural languages. The techniques that we present here are based on different machine learning techniques (kNN and SVM) and a byte-level n-gram document representation, avoiding many of the above mentioned difficulties. Since we use byte-level n-grams to represent the documents, there is no need for any text preprocessing or higher level processing, such as tagging, parsing, or other language dependent and nontrivial natural language processing tasks. The techniques adapted to byte-level n-grams document representation can be thus applied to any text hierarchy without prior knowledge about the language or even about the used coding scheme, which we consider the main novelty of our contribution. In order to demonstrate the effectiveness and language independence of the HTC techniques and to test and compare performance with the corresponding FTC techniques and among themselves, we focus on English and

Chinese as representatives of different language paradigms and the most widely used languages on the Internet[1]. To the best of our knowledge, this has not been done before, using the proposed techniques and document representation. We use five single labeled tree-structured benchmark category hierarchies: Reuters-Hier1, Reuters-Hier2 [25], 15NGHier and 20NGHier [23] in English and TancorpHier in Chinese [37].

The rest of the paper is organized as follows. Sect. 2 introduces related work. Sect. 3 presents kNN and SVM algorithms for the HTC n-gram techniques proposed. Experimental framework is presented in Sect. 4 while Sect. 5 reports on experimental results and comparisons with other BOW SOA (bag-of-words, state-of-the art) methods. In Sect. 6 we give a conclusion.

2. Related Work

Many FTC and HTC techniques have been developed and extensively applied in different areas such as economy [42], medicine [2], news industry [14], oil industry [32], etc. Although there is a number of techniques developed and applied to larger corpora ([10], [4]), we will mention only previously published techniques, applied to the same Reuters and 20-Newsgroups hierarchies in English, and TanCorpHier dataset in Chinese that we used in this article.

The use of the Reuters hierarchies in the field of HTC dates back to at least 1997, when Koller and Sahami [20] proposed the use of a local classifier per parent node approach. Six years later Sun and his colleges [36] presented two top-down level-based hierarchical classification methods based on binary Naïve Bayes (NB) and SVM classifiers. They proposed a new performance measurement framework for hierarchical categorization. In this framework, they incorporate the contributions of misclassified documents into the definition of performance measures. Using the framework, they evaluated the performance of the two HTC techniques using the Reuters hierarchies constructed in a similar way as Koller and Sahami did. They showed that SVM classifiers performed well when there were enough training documents and that extended performance measures could help to better determine the performance of HTC techniques by considering contributions from misclassified documents. Many recent studies used different hierarchical versions of Reuters corpus as well ([18], [39], [15], [24]).

Important results of the TC on English dataset 20-Newsgroups are achieved by Lan et al. [22]. They have investigated several widely-used unsupervised and supervised term weighting methods in combination with SVM and kNN algorithms. They introduced new techniques called tf.rf (term frequency - relevance frequency) which have proved to be more effective than others. Hierarchical version of this corpus was used in [30]. The authors built a two-level hierarchy from 15 categories from this dataset. They showed that the accuracy of the naive Bayes text classifier can be considerably improved by taking advantage of a hierarchy of categories.

The TC problem for languages other than English has been considered as well. Regarding TC focused on Chinese, the TanCorpHier dataset used in this work has been widely used. Authors in [26] have been concentrated on single-label, tree-structured, mandatory leaf-node prediction problem. They proposed an instance-centric HTC frame-

[1] http://www.internetworldstats.com/stats7.htm

work based on decision-theoretic rough set model. Using TanCorpHier dataset they demonstrated that the proposed technique performs better than the FTC and standard HTC. The same authors in [27] proposed blocking distribution based topology reconstruction method for HTC problem. Firstly, they employed blocking distribution recognition technique to mining out the high-level misclassification category. Then, original hierarchical structure is constructed using blocking direction information obtained ahead, which increases the path for the blocking instance to the correct subcategory.

It should be mentioned that the machine learning techniques that we adapted and applied to HTC and specific document representation (kNN and SVM) have been previously successfully extended to handle structured outputs and applied to hierarchical classification in different (non-textual) domains [31], [40].

3. HTC Techniques

Prior to running TC process, a text document has to be transformed from the full text into a document vector which describes the contents of the document. In this paper we chose byte-n-gram based document representation approach. Extracting byte n-grams from a document is like moving an n-byte wide "window" across the document internal representation, byte by byte. Each window position covers n bytes, defining a single n-gram. In the case of English and most other Latin-alphabet languages, character-level and byte-level n-gram models are quite similar due to the fact that one character is usually represented by one byte. The only difference is that character-level n-grams use letters only and typically ignore digits, punctuation, and whitespace while byte-level n-grams use all printing and non-printing characters. In the case of Asian languages, one character is usually represented by two bytes, depending on the coding scheme that is used.

N-grams of bytes and n-grams of characters are equally used for text representation in solving different data mining tasks, with similar results. Although byte n-grams sometimes do not have specific meaning, especially for humans (for example, when they contain only one of two bytes that represent a character), their extraction from a text does not require the information about the used coding scheme, which is why they represent simplified representation for computer processing. Therefore, byte n-grams have been successfully used to represent text in order to solve many problems in the field of natural language processing ([19], [1], [9], [33]).

The use of byte n-grams has a lot of advantages: language independence, relative insensitivity to spelling variations/errors, word stemming is got essentially for free, no linguistic knowledge is required, independence of alphabet, only one pass processing is required end others. The main disadvantage of using n-gram technique is that it yields a large number of n-grams. N-gram techniques have been successfully used for a long time in a wide variety of problems and domains. In natural language processing they turn out to be effective in many applications, for example, text compression [43], spelling error detection and correction [44], information retrieval [7], language identification [38], authorship attribution [19], sentiment polarity detection [13] etc. They also prove useful in domains not related to language processing such as music representation [8], computational immunology [29], protein categorization [35] etc.

3.1. kNN based approach

Flat Text Categorization Technique As a baseline TC technique we use kNN n-gram-based flat text categorization techinque (*kNN_nF*), first used to solve the FTC problem in Serbian, English and Chinese [12] and then to solve the problem of sentiment polarity detection in movie reviews in English and Spanish [13] (the technique was first introduced in [19] to solve the authorship attribution problem).

In *kNN_nF*, category, training and test document profiles are defined as ordered sets of pairs (x_1, f_1), (x_2, f_2),..., (x_L, f_L) of the L most frequent byte n-grams x_i and their normalized frequencies f_i obtained by dividing the number of occurrences of the n-gram x_i with the number of occurrences of all n-grams in the corresponding document. Assigning a document to a category is performed based on (dis)similarity between their two profiles.

Algorithms 1 and 2 give the detailed procedures for training and test phase, respectively, of the flat kNN n-gram-based classifier *kNN_nF*. The dissimilarity measure playing an important role in them.

Algorithm 1 Train_kNN_nF(C,D(Train),n_min, n_max, L_min, L_max, step_L)

Input: Set of category labels C, training set of documents $D(Train)$, initial and final values (with a step) for training classifier parameters: n_min, n_max; L_min, L_max, $step\_L$.
Output: (n, L) which provide the highest accuracy

for each n from n_min to n_max **do**
  //Produce the set of "category documents"
  for each $c \in C$ do
    $doc(c) \leftarrow ConcatenateTextsOfAllTrainingDocsInCategory(D(Train), c)$
  $D(C) \leftarrow \bigcup_{c \in C} doc(c)$
  //For each training document and category document, construct its profile
  for each $doc \in D(Train) \cup D(C)$ **do**
    $Ngrams(doc) \leftarrow ExtractAllByteLevelNgrams(doc, n)$
    for each $x \in Ngrams(doc)$ **do**
      $frequencies[x] \leftarrow CalculateTheNormalizedFrequency(x, doc)$
    $Profile(doc) \leftarrow ListNgramsByDescFreq(\bigcup_{x \in Ngrams(doc)}(x, frequencies[x]))$
  for each L from L_min to L_max with step $step\_L$ **do**
    //Calculate dissimilarity measure between each document and category profile, cut of at the length L
    for each $doc_t \in D(Train)$ **do** $Profile_L(doc_t) \leftarrow Profile(doc_t)|L$
    for each $c \in C$ **do** $Profile_L(doc(c)) \leftarrow Profile(doc(c))|L$
    for each $doc_t \in D(Train)$ **do**
      for each $doc(c) \in D(C)$ **do**
        $diss_{tc} \leftarrow DissimilarityMeasure(Profile_L(doc_t), Profile_L(doc(c)))$
      //Select the most similar category (or categories)
      $c(doc_t) \leftarrow argmin_{c \in C} diss_{tc}$
    Compute the accuracy of the produced categorization
return n and L which provide the highest accuracy

Definition 1. *Dissimilarity measure d is a function that maps the Cartesian product of a set of profiles Π into the set of positive real numbers R. Symbolically, $d : \Pi \times \Pi \longrightarrow R$. A metric dissimilarity d satisfies the following: nonnegativity, reflexivity, symmetry and triangle inequality [11].*

Algorithm 2 Test_kNN_nF(doc, n, L, $\{Profile_L(doc(c)), c \in C\}$)

Input: Test document $doc \in D(Test)$, (n, L) obtained from the training phase presented in Algorithm 1, set of category profiles $\{Profile_L(doc(c))\}$ for all the category documents $doc(c)$ and the trained n and L
Output: Category (or rarely categories) predicted by classifier
1: //Construct test document profile, $Profile(doc)$, for the given n (steps from 8 to 11 in Algorithm 1)
2: //Cut off obtained test document profile at the length of the given profile length L
3: $Profile_L(doc) \leftarrow Profile(doc)|L$
4: **for each** $c \in C$ **do**
5: $\quad diss\_c \leftarrow DissimilarityMeasure(Profile_L(doc), Profile_L(doc(c)))$
6: //Select the most similar category (or categories)
7: $c\_predicted[] \leftarrow argmin_{c \in C} diss_c$
8: **return** $c\_predicted[]$

In this paper we used dissimilarity measure presented by Kešelj [19] which proved to be the best choice for TC considering 19 different dissimilarity measures presented in [38]. This measure has a form of relative distance:

$$d(\mathcal{P}_1, \mathcal{P}_2) = \sum_{x \in \mathcal{P}_1 \cup \mathcal{P}_2} \Big(\frac{2 \cdot (f_1(x) - f_2(x))}{f_1(x) + f_2(x)} \Big)^2, \tag{1}$$

where $f_1(x)$ and $f_2(x)$ are frequencies of an n-gram x in the category profile $\mathcal{P}_1$ and the document profile $\mathcal{P}_2$, respectively.

Hierarchical Text Categorization Technique We present a multi label tree-structured top-down HTC method with mandatory leaf node prediction using k nearest neighbors (kNN) classifiers, with $k = 1$. In a top-down HTC approach, there is one or more classifiers at each level of the category tree (for each internal non-leaf node we build a local classifier that works as a flat classifier for its child categories). This approach is known as *local classifier per parent node* approach. Since the hierarchical category structure is known in advance, we actually decompose the categorization problem into a set of sub-problems corresponding to hierarchical splits in the category tree. The whole HTC process is achieved with the cooperation of all constructed internal node local classifiers that solve corresponding sub-problems. In the test phase, the document is first classified by the classifier at the root level into one or more $level - 1$ categories. It is then further classified by the one or more classifiers at the current level category(ies) until it reaches one or more final categories. In the case of mandatory leaf node prediction, these final categories need to be leaves.

Since HTC process is accomplished with the cooperation of all local classifiers at each level and each internal node of the hierarchy, it is important to properly choose how to construct the internal node classifiers in order to guarantee the maximum accuracy for the whole HTC process. For a given internal node, local classifier works as a flat classifier for its child categories. In order to achieve maximum accuracy we need to find the best local classifier for each internal node. Note that effectiveness of the presented basic *kNN_nF* can be controlled by two parameters: n-gram size n and profile length L. Finding the best classifier for internal node thus means finding the best choice for classifier parameters n and L. Alogrithm 3 and Algorithm 4 present the algorithms for training and test phase of kNN byte n-gram-based hierarchical text categorization technique (*kNN_nH*).

One of the obvious problems with *kNN_nH* as a top-down approach is that a misclassification at a parent category may force a document to be discarded before it can be classified into the child categories. This is usually known as the "blocking" problem [36]. This gives the opportunity to strengthen and improve this technique by employing some error recovery solutions.

Algorithm 3 Train_kNN_nH(D(Train), T(V,E), TrainingParameters)

Input: Training set $D(Train)$, a category tree $T(V = \{I \cup L\}, E)$ with the height H, where I, L are sets of internal and leaf nodes, respectively, E is set of edges, $TrainingParameters$ is the set of initial and final values (with a step) for training classifier parameters n and L for each internal node
Output: Set of internal node classifiers $ClassifierSet(I)$
1: //For the root node r, find (n, L) which produce the best accuracy for flat categorization to roots child categories
2: $(n_r, L_r) \leftarrow Train\_kNN\_nF(Children(r), D(Train), n\_r\_min, n\_r\_max, L\_r\_min, L\_r\_max, step\_L\_r)$
3: **for each** $c_i \in I \setminus \{r\}$ **do**
4: //Find L which produce the best accuracy for flat categorization to c_i's child categories
5: $(n_r, L_{c_i}) \leftarrow Train\_kNN\_nF(Children(c_i), D(Train_{Children(c_i)}), n_r, n_r, L\_i\_min, L\_i\_max, step\_L\_i)$
6: $ClassifierSet(I) \leftarrow \bigcup_{c_i \in I}(n_r, L_{c_i})$
7: **return** $ClassifierSet(I)$

Algorithm 4 $Test\_kNN\_nH(doc, T(V, E), ClassifierSet(I))$

Input: Test document $doc \in D(Test)$, a category tree $T(V = \{I \cup L\}, E)$ with the height H, where I, L are sets of internal and leaf nodes, respectively, E is set of edges, $ClassifierSet(I)$ is set of internal node classifiers
Output: Category (or rarely categories) predicted by classifier
1: $h = 0$; $c\_predicted\_current[] = \{c\_root\}$
2: // From the root down to the leaf node parent
3: **while** $h <= H - 1$ **do**
4: **for each** $c\_h \in c\_predicted\_current[]$ **do**
5: // Choose the corresponding classifier parameters obtained in the training phase for c_h
6: $(n_{c\_h}, L_{c\_h}) \leftarrow ClassifierSet(c\_h)$
7: **let** $c\_predicted\_h'[] \in Children(c\_h)$ be the categories (at the level $h' = h + 1$) assigned to doc
8: // from the set $c\_predicted\_current[]$ exclude category c_h and add all categories from the $c\_predicted\_h'[]$
9: $c\_predicted\_current[] = (c\_predicted\_current[]/\{c\_h\}) \cup c\_predicted\_h'[]$;
10: h++;
11: // return all predicted leaf categories
12: return $c\_predicted\_current[]$

3.2. *SVM struct* based approach

Support vector machine (SVM) classifiers have been shown to be efficient and effective for TC. Here we use the *SVM struct* ([40]) framework which represents a generalization of SVM algorithm for structural output (sequences, trees, directed acyclic graphs, etc). We adjusted the original *SVM struct* algorithm in two ways: first, to act as a multiclassifier, following the flat approach (*SVM_nF*), and second, to predict a category as a path in the category ontology, ending in leaves, following the big-bang approach (*SVM_nH*).

In both techniques, each text document is represented as a sequence of byte level n-grams. In order to determine how important a byte n-gram x is to a text document d in a dataset, we use term frequency-inverse document frequency (*tf-idf*) numerical statistic as

a weighting factor. Various ways for determining the exact values of tf and idf statistics exist [28]. In this paper, we used the following 4 tf-idf statistics:

1. *classic tf-idf*: $tf \cdot \log \frac{N}{1+n_x}$
2. *log tf-idf*: $(1 + \log(tf)) \cdot \log \frac{N}{1+n_x}$
3. *boolean 1 tf-idf*: $\log \frac{N}{1+n_x}$
4. *boolean 2 tf-idf*: $\log (1 + \frac{N}{n_x})$

where tf represents normalized frequency of the n-gram in the document, N is the total number of documents in the dataset and n_x is the number of documents in which the n-gram x appears.

Before presenting the *SVM_nF* and *SVM_nH* techniques we provide a brief overview of binary and structured SVM categorization. A standard approach in training predictors for binary classification problems is to learn a discriminant function $F(\boldsymbol{x})$ and classify the input $\boldsymbol{x}$ according to the sign of $F(\boldsymbol{x})$. Since linear methods usually have efficient training algorithms, it is common to assume that the discriminant function is linear: $F(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle$, where $\boldsymbol{w}$ is a vector of parameters. Input data $\boldsymbol{x}$ can be mapped into another feature space using a function $\Psi(x)$ which turns the discriminant function into the function $F(\boldsymbol{x}) = \langle \boldsymbol{w}, \Psi(\boldsymbol{x}) \rangle$.

A binary classifier can predict whether a text belongs to a specified category or not. In order to predict the category the text belongs to, we turn to structured output learning. In this generalization, the discriminant function becomes a function of both inputs and outputs, $F(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{y}$ can be any structured output: tree, sequence, directed acyclic graph, etc. Here, the discriminant function represents the level of compatibility of the input $\boldsymbol{x}$ with the output $\boldsymbol{y}$. If X denotes the input space and Y the output space, structured output methods infer a label from the following equation:

$$\hat{y} = \underset{\boldsymbol{y} \in Y}{\operatorname{argmax}} (F(\boldsymbol{x}, \boldsymbol{y})).$$

Again, we assume that F is linear in $\boldsymbol{w}$, $F(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{w}, \Psi(\boldsymbol{x}, \boldsymbol{y}) \rangle$ in some space defined by the mapping Ψ. Unlike the standard SVMs, in structural-output setting the function Ψ is a joint function of both inputs and outputs.

In this paper, we employed the so-called 1-slack formulation of the problem with margin rescaling ([17]):

$$\min_{\boldsymbol{w}, \boldsymbol{\xi}} \frac{||\boldsymbol{w}||^2}{2} + C\xi$$

$$s.t. \forall (\bar{\boldsymbol{y}}_{\mathbf{1}}, \ldots, \bar{\boldsymbol{y}}_{\boldsymbol{N}}) \in Y^N : \frac{1}{N} \langle \boldsymbol{w}, \sum_{i=1}^{N} [\Psi(\boldsymbol{x_i}, \boldsymbol{y_i}) - \Psi(\boldsymbol{x_i}, \bar{\boldsymbol{y}}_{\boldsymbol{i}})] \rangle \geq \frac{1}{N} \sum_{i=1}^{N} \Delta(\boldsymbol{y_i}, \bar{\boldsymbol{y}}_{\boldsymbol{i}}) - \xi.$$

Just like in classic SVM, we allow for certain training examples to be misclassified, which is penalized by variable ξ. Parameter C regulates the influence of the misclassification on the objective function and its value is tuned during the validation process. Function Δ represents loss function between two outputs. We used *SVM struct* framework with the cutting plane algorithm as an underlying optimizer ([17]). The details of the algorithm can be found in original papers ([40],[41],[17]).

In the first approach (*SVM_nF*), we adjusted original *SVM struct* to work as a multi-classifier. Input set X consists of vectors that contain values of the selected tf-idf statistics (*classic, log, boolean1* or *boolean2*) for each n-gram in training text, for selected length $n \in \{2,3,4,5,6,7,8\}$ of byte n-grams. Each text is represented by one such vector, and for different tf-idf statistics used and n-gram length, there are 28 variations of representation for each corpus (7 for each of 4 tf-idf statistics: $tf - idf = classic, n = 2, \ldots, n = 8, \ldots, tf - idf = boolean2, n = 2, \ldots, n = 8$). The dimension of each input vector is p, the total number of different n-grams in the training set. Output set Y consists of integers from 1 to q, where q is the total number of categories. Function Ψ was constructed as a vector of dimension $p \cdot q$. That way, each category obtained a block in the joint representation with zero values, if the text does not belong to that category, or values equal to the input vector x, otherwise. Here is an example of the function Ψ for the text $\boldsymbol{x}$ which belongs to the category k (x_i is the number of occurrences of the i-th n-gram in the text):

$$\Psi(\boldsymbol{x}, k) = [\ \underbrace{0, \ldots, 0}_{\text{block for class 1}}\ , \ldots, 0, \ldots, 0, \underbrace{x_1, \ldots, x_p}_{\text{block for class k}}, 0, \ldots, 0, \ldots, \underbrace{0, \ldots, 0}_{\text{block for class q}}\ ].$$

In the second approach (*SVM_nH*), we exploited the ontology of each corpus and considered each category as its path beginning in the ontology's root and ending in one of its leaves. Input set X is equivalent to the one in the first approach whereas output set Y is the set of sparse vectors y, which dimension is equal to the number of nodes in the specific ontology. If a category contains a certain node in its graphical representation, the value of the vector y will be 1 on the position corresponding to that node, zero otherwise. For example, if we enumerate the nodes of the Reuters-Hier1 ontology in the following way: *corn*-1, *wheat*-2, *ship*-3, *natural gas*-4, *grain*-5, *crude oil*-6, *root*-7 (see Fig. 1), then the category *corn* can be described as $[1, 0, 0, 0, 1, 0, 1]$. Here, the dimension of the function Ψ is equal to the total number of nodes in the considered ontology multiplied by the number of different byte n-grams. Analogously, each node obtained its block in the joint representation, with zero values, if the category that the text belongs to does not contain that node, or values equal to the input vector x otherwise. Here is an example of function Ψ for text $\boldsymbol{x}$ which belongs to a category 1:

$$\Psi(\boldsymbol{x}, 1) = [\underbrace{x_1, \ldots, x_p}_{\text{block for node 1}}, 0, \ldots, 0, \underbrace{x_1, \ldots, x_p}_{\text{block for node 5}}, 0, \ldots, 0, \underbrace{x_1, \ldots, x_p}_{\text{block for node 7}}].$$

Since in this particular application the cardinality of Y is not large, the argmax for prediction and for separation oracle ([17]) in both *SVM_nF* and *SVM_nH* techniques were computed by explicit enumeration. In the opposite case, if the cardinality had been large, more sophisticated algorithms would have had to be exploited.

Algorithm implementations for both kNN and SVM, flat and hierarchical techniques, can be obtained on request from the authors.

4. Experimental Settings

4.1. Data Collections

In order to test presented HTC techniques, we used five tree-structured banchmark datasets: Reuters-Hier1, Reuters-Hier2, 15NGHier and 20NGHier in English and TancorpHier in

Chinese. All text documents can only be assigned to leaf categories and every leaf node has the same height.

Reuters-Hier1 and Reuters-Hier2 Reuters corpus is one of the most popular datasets used in TC [25]. Much work in HTC have also used this collection. The Reuters corpus has 21,578 text documents labelled by 135 categories which are not organized in hierarchical manner. To conduct the experiment, category trees needed to be manually derived. We have derived two category trees from the Reuters-21578 dataset using the same structure of the trees as Sun and others in [36] (see Fig. 1) and the same approach of extracting trees as Koller and his colleges in [20]. In our hierarchies every document contained one (and only one) major topic and one (and only one) minor topic (or subtopic). For example, in the Reuters-Hier1 corpus document belongs to the category "corn" if the topic of that document is set to "corn" and "grain" but is not set to "crude oil", "wheat", "ship", or "natural gas". The minor topics are grouped together to the major topics, and major topics are all grouped together at the top level of the hierarchy. For example, all documents that belong to the "corn" or "wheat", at the same time belong to the "grain" category. In the classification process, we used only title and body parts of the Reuters articles. The major and minor topics in the two hierarchies are described in Table 1. We used Lewis Split to split the Reuters collection into training and test sets (as has been done in [36]). These hierarchies represent only small percentage of the entire Reuters corpus, and we made them publicly available[2].

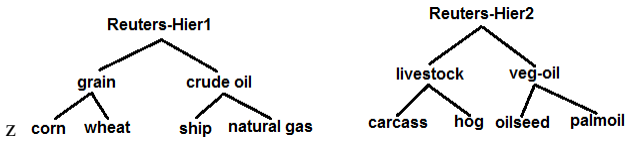

Fig. 1. Reuters hierarchical trees used in our experiments

Table 1. Number of training and test documents for Reuters hierarchies

Reuters-Hier1				**Reuters-Hier2**			
Major topic	**Minor topic**	**Dataset size**		**Major topic**	**Minor topic**	**Dataset size**	
		Training	**Testing**			**Training**	**Testing**
grain	corn	121	36	livestock	carcass	37	10
	wheat	147	50		hog	12	6
crude	ship	41	38	veg-oil	oilseed	17	10
	nat-gas	51	18		palm-oil	26	10
Total		**360**	**142**	**Total**		**92**	**36**

[2] Available at www.matf.bg.ac.rs/~jgraovac/HTC/ReutersHierarchies.zip

15NGHier and 20NGHier The 20-Newsgroups[3] is a collection of approximately 20000 newsgroup documents, evenly divided into 20 different newsgroups, each corresponding to a different topic. It was first collected by Lang [23]. Three versions of this dataset are publicly available but the most popular is bydate version that is sorted by date into training (60%) and testing (40%) sets. This is the corpus edition we used for the TC technique presented. 20-Newsgroups corpus is a single label dataset so each document is guaranteed to have only one category label. There are 18846 documents in this corpus.

According to subject matter, we divided this corpus into 2 hierarchical levels. The first level consists of 6 categories and the second consists of 20 subcategories (see figure 2 (right)). We called this hierarchy 20NGHier. In [30] the authors built a two-level hierarchy from 15 categories that fit into 5 top level categories (see figure 2 (left)), so we called this hierarchy 15NGHier. We used both hierarchies in our work.

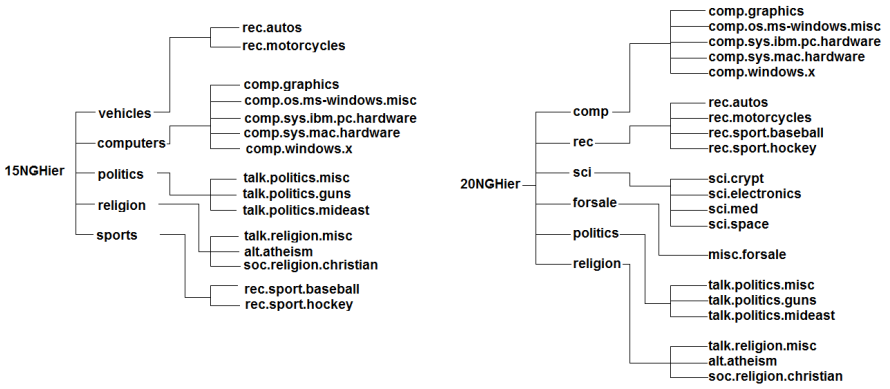

Fig. 2. 20-Newsgroups hierarchical trees used in our experiments

TanCorpHier The TanCorpHier corpus[4] is a collection of 14,150 text documents in Chinese collected by Songbo Tan [37]. The corpus is divided in two hierarchical levels. The first level contains 12 large categories (major topics) (art, car, career, computer, economy, education, entertainment, estate, medical, region, science and sport) and the second consists of 60 subclasses (minor topics). This corpus can serve as three categorization datasets: one hierarchical dataset (TanCorpHier) and two flat datasets (TanCorp-12 and TanCorp-60). To evaluate the proposed HTC techniques, we conducted experiments on the hierarchical structure. In our work, documents are randomly sampled for training set and test set in the ratio $2:1$.

4.2. Evaluation Metrics

To analyze the performance of the HTC, we used the typical evaluation metrics for FTC: Precision (P), Recall (R), and F1 measure [3]. All these measures can be aggregated over

[3] 20-Newsgroups: people.csail.mit.edu/jrennie/20Newsgroups/

[4] Available at http://www.searchforum.org.cn/tansongbo/corpus.htm

all categories in two ways: micro-averaging – the global calculation of measure considering all the documents as a single dataset regardless of categories, and macro-averaging – the average on measure scores of all the categories. In this paper we used micro-averaged F1 measure (mi-F1).

Most researchers used these standard FTC evaluation measures for HTC. Main disadvantage of these measures are penalizing the errors at different levels of the hierarchy in the same way so some authors have proposed their own HTC evaluation measures ([36], [5], [21]).

5. Experimental Results

Here we present the results obtained by kNN and SVM techniques (flat and hierarchical). We compare the results obtained for different granularity of n-grams (byte, character and word level) in document representation, then we present results of statistical comparison of the techniques performance and compare techniques with the previously published BOW SOA results and finally give a brief discussion. In the case of both kNN and SVM approaches, all document processing is done on the training set only so we avoid creating overly optimistic experimental results from having any prior access to the test data.

5.1. kNN techniques

As an accuracy baseline, we ran the *kNN_nF* where each minor topic is treated as a separate category. We trained classifier parameters n and L (using Algorithm 1) for $n = 3$ to 8 and $L = 100$ to 5000 with step 100 in the case of Reuters datasets, $L = 10000$ to 600000 with step 10000 in the case of 15NGHier and 20NGHier and $L = 1000$ to 10000 with step 1000 in the case of TancorpHier dataset. The best results are obtained for $n = 3$, $L = 4600$; $n = 3$, $L = 3400$; $n = 7$, $L = 250000$; $n = 8$, $L = 300000$ and $n = 8$, $L = 10000$ for Reuters-Hier1, Reuters-Hier2, 15NGHier, 20NGHier and TancorpHier datasets, respectively (see Table 2).

Then we ran top-down *kNN_nH*. As we already mentioned in the Sect. 3.1, test documents are classified in the hierarchy by filtering them through the first level classifier and then sending the document down to the chosen internal node on the second level where a final category assignment (into a minor topic) is made. Note that errors made at the first level of the hierarchy are unrecoverable at the second level. Thus, our technique needs to make two correct predictions in order for a test document to be considered properly classified. At first, we trained classifier parameters (using Algorithm 3) for $n = 3$ to 8 and $L = 1000$ to 10000 with step 1000 at the first level and $L = 100$ to 5000 with step 100 at the second level of hierarchy in the case of Reuters datasets; for $n = 3$ to 8 and $L = 10000$ to 600000 with step 10000 at the both levels of hierarchy in the case of 15NGHier and 20NGHier datasets and in the case of Tancorp hierarchy, we trained parameters for $n = 3$ to 8 and $L = 10000$ to 100000 with step 10000 at the first level and $L = 1000$ to $L = 10000$ with step 1000 at the second level of hierarchy. Best values for n and L for the flat and hierarchical kNN (at the first and second level of the hierarchies), as well as the results obtained on the test datasets for the corresponding classifeirs, for all five hierarchies, are presented in the Table 2.

Table 2. Mi-F1 results for *kNN_nF* and *kNN_nH* learning (all the values are in percentage). Best values for n and L for the flat kNN and the hierarchical kNN (at the first level and each of the nodes at the second level of the hierarchies), as well as the results obtained on the test datasets for the corresponding classifiers, for all the three corpora, are presented. Considerably better results between flat and hierarchical approach, for each dataset, are marked with boldface

Methods	Datasets														
	Reuters-Hier1			Reuters-Hier2			15NGHier			20NGHier			TancorpHier		
	n	L	Mi-F1	n	L	Mi-F1	n	L	Mi-F1	n	L	Mi-F1	n	L	Mi-F1
kNN_nF	3	4600	85.21	3	3400	**83.33**	7	250000	**82.57**	8	300000	**81.61**	8	10000	**79.77**
kNN_nH	3	7000 2600 (grain) 4600 (crude oil)	**87.32**	3	3000 2600 (livestock) 4600 (veg-oil)	**83.33**	7	550000 300000 (vehicles) 250000 (computers) 420000 (politics) 310000 (religion) 310000 (sports)	**82.62**	7	270000 250000 (comp) 300000 (rec) 300000 (sci) 270000 (forsale) 420000 (politics) 310000 (religion)	80.08	5	100000 10000 (for all nodes)	78.14

5.2. SVM techniques

As described in Chapter 3.2, for each corpus totally 28 representations were generated, for 4 different tf-idf statistics (*classic,log,boolean1,boolean2*) and for 7 lengths of n-grams ($n \in \{2, 3, 4, 5, 6, 7, 8\}$). Both types of *SVM struct* classifiers (flat and hierarchical) were applied on each representation for all corpora in the following way:

1. In order to tune the value of parameter C and two metaparameters: the type of tf-idf statistics and n-gram length, we performed 10-fold cross-validation. For the purpose of validation, the training set was divided into 10 parts of approximately equal size. We trained models with different combinations of parameters on $\frac{9}{10}$ of the training set and tested it on the remaining $\frac{1}{10}$ and repeated this process ten times for different tenth of the training set aside. We used the average value of the obtained results for each combination of parameters. Since corpora Reuters 1 and Reuters 2 are small, cross-validation was performed 10 times on randomly shuffled dataset, and the results obtained were again averaged. For each corpus, the validation set was separated from the test set.
2. For optimal values of method's parameters, classification model was trained on the entire training set
3. Classification model has been tested and evaluated on the test set

Validation results for both flat and hierarchical SVM struct classifier for all corpora can be found online[5]. Table 3 displays the evaluation of the selected models (with parameters tuned on the validation set) on the test set for each corpus. Overall, best result obtained with *SVM struct* is for Tancorp corpus, hierarchical type of classifier, *boolean 2* tf-idf statistics and n-gram lenght 3. Tf-idf statistics that showed best performance is *boolean 2* which was selected for 5 out of 6 proposed results (*boolean 1* was selected for flat classifier for Reuters 2 corpus). n-gram lengths that showed best performances are 4 (Reuters 1, Reuters 2 for flat classifier) and 3 (Tancorp, Reuters 2 for hierarchical classifier).

[5] `http://www.matf.bg.ac.rs/~jovana/HTC/htc_complete_results.xlsx`

Table 3. Mi-F1 results of flat (*SVM_nF*) and hierarchical (*SVM_nH*) SVM clasifier (all the values are in percentage). Considerably better results between flat and hierarchical approach, for each dataset, are marked with boldface

Methods	Datasets														
	Reuters-Hier1			Reuters-Hier2			15NGHier			20NGHier			TancorpHier		
	n	tf-idf	Mi-F1	n	tf-idf	Mi-F1	n	tf-idf	Mi-F1	n	tf-idf	Mi-F1	n	tf-idf	Mi-F1
SVM_nF	4	boolean2	**85.92**	4	boolean1	75	4	boolean	**86.63**	4	boolean	**85.40**	3	boolean2	**86.73**
SVM_nH	4	boolean2	**85.21**	3	boolean2	**80.56**	5	boolean	84.04	4	boolean2	**85.37**	3	boolean2	**87.04**

5.3. Byte-level n-grams vs. character and word level n-grams

In order to make the comparison between byte and character or word n-gram based document representations, we conducted additional experiments for character and word n-grams in the case of Reuters-Hier1 dataset, for flat and hierarchical kNN and SVM techniques. Results are presented in Table 4. In the case of byte and char n-grams we obtained the best results for 4-grams (in the case of SVM) and 3-grams (in the case of kNN), while in the case of word n-grams, the winner is 1-gram document representation (BOW) for both techniques, SVM and kNN. From this table we can see that the best results are obtained for byte n-grams (in the case of kNN techniques) and for word n-grams (in the case of SVM techniques). As we already mentioned, in this paper we used byte n-grams, because they ensure full language independence of the techniques.

Table 4. Mi-F1 results (in percentages) of flat and hierarchical kNN and SVM for the Reuters-Hier1 corpus. Best results between byte, character and word n-grams are marked with boldface

Mi-F1	***kNN_nF***	***kNN_nH***	***SVM_nF***	***SVM_nH***
byte n-grams	**85.21**	**87.32**	85.92	85.21
char n-grams	83.10	85.21	83.10	85.21
word n-grams	82.39	81.69	**88.03**	**85.92**

5.4. Comparison with the previous state-of-the-art results

In order to evaluate the presented results, we compare them, based on mi-F1 measure, with the results published by other authors, obtained by different techniques, using the same datasets. In the case of English Reuters hierarchies we compare results with the Sun's [36] results. We used the same structure of Reuters hierarchies and similar procedure for constructing the hierarchies. In the case of 20-Newsgroups corpus, for comparison purposes we used reported BOW kNN and SVM results cited in [22], while in the case of 15NGHier, as the baseline for our comparison we used results of McCallum et al. [30]. In the case of TanCorpHier dataset, we compare our results with the results of Li et al. [26]. Comparison results are presented in Table 5. In case of the smallest ReutersHier2 corpus, both kNN and SVM hierarchical techniques outperform earlier published ones, while *kNN_nH* performs the best of all on the smallest ReutersHier2 corpus. Additionally, *SVM_nH* outperforms current results for the largest TanCorpHier corpus, while *SVM_nF* outperforms current results for the 15NGHier and 20NGHier datasets.

Table 5. Mi-F1 comparison of the kNN and SVM techniques with the previous results (all the values are in percentage)

Authors	Technique	Datasets	Mi-F1
Sun (2003)	BOW SVM	Reuters-Hier1	**88.59**
		Reuters-Hier2	73.41
	BOW NB	Reuters-Hier1	76.54
		Reuters-Hier2	65.97
McCallum et al. (1998)	BOW NB	15NGHier	84
Lan et al. (2009)	BOW SVM	20NGHier	80.81
	BOW kNN		69.1
Li et al. (2010)	BOW SVM	TanCorpHier	82.45
Our proposal	kNN_nH / kNN_nF (top-down approach)	Reuters-Hier1	87.32 (-1.27) / 85.21 (-3.38)
		Reuters-Hier2	**83.33 (+9.92) / 83.33 (+9.92)**
		15NGHier	82.76 (-1.24) / 82.57 (-1.43)
		20NGHier	81.61 (+0.8) / 80.08 (-0.73)
		TanCorpHier	78.14 (-4.31) / 79.77 (-2.68)
	SVM_nH / SVM_nF (big-bang approach)	Reuters-Hier1	85.21 (-3.38) / 85.92 (-2.67)
		Reuters-Hier2	80.56 (+7.15) / 75 (+1.59)
		15NGHier	84.04 (+0.04) / **86.63 (+2.63)**
		20NGHier	85.37 (+4.56) / **85.40 (+4.69)**
		TanCorpHier	**87.04 (+4.59)** / 86.73 (+4.28)

Note: With boldface we marked the best results obtained for each dataset. For our proposal, in brackets we presented differences between our results and the best Sun's [36], Lan's [22], McCallum's [30] and Li's [26] results obtained for corresponding data collection.

5.5. Discussion

N-gram based text representation, either byte-level, character-level or word-level, depending on granularity, proves useful in application of different text classification methods. Although it seems not to be a single n that performs the best, n-gram size can be efficiently trained – or chosen – by different machine learning methods, on specific corpora in specific languages, so as to produce the state-of-the-art results.

Although trained parameters differ for different methods and corpora, results of the corresponding best classifiers for each corpus and each method (kNN and SVM, flat and hierarchical) are compared with other relevant classification results and among themselves. The proposed methods achieved state-of-the-art results for Reuters-Hier2 (*kNN_nH*), 15NGHier (*SVM_nF*), 20NGHier (*SVM_nF*) and TanCorpHier (*SVM_nH*) datasets. The results presented show higher accuracy of *kNN_nH* over *SVM_nH* for Reuters-Hier1 and Reuters-Hier2 datasets, whereas *SVM_nH* performs more accurately than *kNN_nH* for 15NGHier, 20NGHier and TanCorpHier datasets. The reason for this may be the fact that 15NGHier, 20NGHier and TanCorpHier datasets are much larger than Reuters hierar-

chies so that more sophisticated SVM-with-structural-output method is better trained on such corpora. Furthermore, flat kNN variant outperforms hierarchical one on TancorpHier dataset. Again, the reason for this may be the fact that TancorpHier corpus, although very large, has a very simple and shallow hierarchy so that hierarchical methods cannot be fully employed. Also, the reason may be suffering from the "blocking" problem. Finally, flat and hierarchical SVM methods do not perform substantially different for the largest TanCorpHier and 20NGHier corpora. The reason for this may be shallow (2-level) hierarchy that prevents this powerful mechanism from riching its best performance.

Overall, we may conclude that the question of whether HTC approach is better than FTC approach remains an open question depending on many factors. One of them is the evaluation measure used. In TC process it does not only matter how many mistakes are made, but also how serious they are. This suggests that, since HTC approaches show similar or better results against the FTC approach when using flat categorization evaluation metrics, they may produce much better results if a hierarchical categorization evaluation measure was used instead [34].

6. Conclusion and Future Work

We presented two new hierarchical text categorization (HTC) techniques based on kNN and SVM machine learning techniques. Due to the byte n-gram based document representation, presented HTC techniques do not require word or character segmentation, do not need any text preprocessing or higher level processing, such as tagging, parsing, or other language-dependent and non-trivial natural language processing tasks. For conducting experiments, we used five tree-structured benchmark datasets: Reuters-Hier1, Reuters-Hier2, 15NGHier and 20NGHier in English and TanCorpHier in Chinese.

In the case of kNN we developed a byte n-gram-based top-down HTC technique (*kNN_nH*). Experimental results confirm that this technique in the case of (smaller) Reuters hierarchies and 15NGHier gives better or equal results than flat text categorization technique, but in the case of (larger) 20NGHier and TanCorpHier it gives lower accuracy. The reason for the latter may be that the HTC technique developed may suffer from the "blocking" problem so it may be improved by incorporating some error recovery mechanism to correct the mistakes made by the parent classifier. This will be our task for future work.

In the case of SVM approach we used *SVM struct* in two ways: first, to act as a flat multiclassifier (*SVM_nF*), and second, to act as a hierarchy classifier (*SVM_nH*) following the big-bang approach. In the case of TanCorpHier, 15NGHier and 20NGHier datasets, we obtained the state-of-the-art results. Since SVM technique is sensitive to the size of the dataset, we obtained lower performance for Reuters hierarchies. We need to stress out that we applied the SVM technique without any feature selection (FS) or feature extraction (FE) technique, which gives this technique great potential for improvements. We plan to test the technique for combination of different size n-grams (for example, 3-grams and 4-grams), for different weight measures (not only the basic tf-idf), for different kernel functions and different FS or FE techniques.

The presented techniques may be extended in different ways. First, different hierarchical evaluation metrics may be applied in order to give different weights to successful classification at different levels of the hierarchy. Also, we plan to test the technique on

more complex hierarchies (directed acyclic graph, non mandatory leaf-node prediction and multi-label case). We expect that the *SVM_nH* will be superior in that case. Another possible research direction would be combining both kNN and SVM approaches in order to improve accuracy. Since our techniques are topic and language independent, we plan to apply them to other domains and languages as well.

Acknowledgments. The work presented has been financially supported by the Ministry of Science and Technological Development, Republic of Serbia, through Projects No. 174021 and No. III47003.

References

1. Abou-Assaleh, T., Cercone, N., Keselj, V., Sweidan, R.: N-gram-based detection of new malicious code. In: Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International. vol. 2, pp. 41–42. IEEE (2004)
2. Adeva, J.G., Atxa, J.P., Carrillo, M.U., Zengotitabengoa, E.A.: Automatic text classification to support systematic reviews in medicine. Expert Systems with Applications 41(4), 1498–1508 (2014)
3. Baeza-Yates, R., Ribeiro-Neto, B., et al.: Modern information retrieval, vol. 463. ACM press New York (1999)
4. Bi, W., Kwok, J.T.: Mandatory leaf node prediction in hierarchical multilabel classification. IEEE transactions on neural networks and learning systems 25(12), 2275–2287 (2014)
5. Costa, E., Lorena, A., Carvalho, A., Freitas, A.: A review of performance evaluation measures for hierarchical classifiers. In: Evaluation Methods for Machine Learning II: papers from the AAAI-2007 Workshop. pp. 1–6 (2007)
6. Damashek, M.: Gauging similarity with n-grams: Language-independent categorization of text. Science 267(5199), 843 (1995)
7. De Heer, T.: Experiments with syntactic traces in information retrieval. Information Storage and Retrieval 10(3-4), 133–144 (1974)
8. Downie, J.S.: Evaluating a simple approach to music information retrieval: Conceiving melodic n-grams as text. Ph.D. thesis, The University of Western Ontario London (1999)
9. Frantzeskou, G., Stamatatos, E., Gritzalis, S., Chaski, C.E., Howald, B.S.: Identifying authorship by byte-level n-grams: The source code author profile (scap) method. International Journal of Digital Evidence 6(1), 1–18 (2007)
10. Gopal, S., Yang, Y.: Hierarchical bayesian inference and recursive regularization for large-scale classification. ACM Transactions on Knowledge Discovery from Data (TKDD) 9(3), 18 (2015)
11. Goshtasby, A.A.: Similarity and dissimilarity measures. In: Image registration, pp. 7–66. Springer (2012)
12. Graovac, J.: A variant of n-gram based language-independent text categorization. Intelligent Data Analysis 18(4), 677–695 (2014)
13. Graovac, J., Pavlovic-Lazetic, G.: Language-independent sentiment polarity detection in movie reviews: a case study of english and spanish. In: 6th International Conference ICT Innovations. pp. 13–22 (2014)
14. Ha-Thuc, V., Renders, J.M.: Large-scale hierarchical text classification without labelled data. In: Proceedings of the fourth ACM international conference on Web search and data mining. pp. 685–694. ACM (2011)
15. Hernández, J., Sucar, L.E., Morales, E.F.: Multidimensional hierarchical classification. Expert Systems with Applications 41(17), 7671–7677 (2014)
16. Joachims, T.: Learning to classify text using support vector machines: Methods, theory and algorithms. Kluwer Academic Publishers (2002)

17. Joachims, T., Finley, T., Yu, C.N.J.: Cutting-plane training of structural svms. Machine Learning 77(1), 27–59 (2009)
18. Ju, Q., Moschitti, A., Johansson, R.: Learning to rank from structures in hierarchical text classification. In: European Conference on Information Retrieval. pp. 183–194. Springer (2013)
19. Kešelj, V., Peng, F., Cercone, N., Thomas, C.: N-gram-based author profiles for authorship attribution. In: Proceedings of the conference pacific association for computational linguistics, PACLING. vol. 3, pp. 255–264 (2003)
20. Koller, D., Sahami, M.: Hierarchically classifying documents using very few words (1997)
21. Kosmopoulos, A., Partalas, I., Gaussier, E., Paliouras, G., Androutsopoulos, I.: Evaluation measures for hierarchical classification: a unified view and novel approaches. Data Mining and Knowledge Discovery 29(3), 820–865 (2015)
22. Lan, M., Tan, C.L., Su, J., Lu, Y.: Supervised and traditional term weighting methods for automatic text categorization. IEEE transactions on pattern analysis and machine intelligence 31(4), 721–735 (2009)
23. Lang, K.: Newsweeder: Learning to filter netnews. In: Proceedings of the 12th international conference on machine learning. pp. 331–339 (1995)
24. Levatić, J., Kocev, D., Džeroski, S.: The importance of the label hierarchy in hierarchical multi-label classification. Journal of Intelligent Information Systems 45(2), 247–271 (2015)
25. Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: Rcv1: A new benchmark collection for text categorization research. Journal of machine learning research 5(Apr), 361–397 (2004)
26. Li, W., Miao, D., Wang, W., Zhang, N.: Hierarchical rough decision theoretic framework for text classification. In: Cognitive Informatics (ICCI), 2010 9th IEEE International Conference on. pp. 484–489. IEEE (2010)
27. Li, W., Wang, W., Chai, L.: Blocking distribution based hierarchical reconstruction for text categorization. Journal of Applied Sciences 13(11), 2123 (2013)
28. Manning, C.D., Raghavan, P., Schütze, H.: Scoring, term weighting and the vector space model. Introduction to Information Retrieval 100, 2–4 (2008)
29. Marceau, C.: Characterizing the behavior of a program using multiple-length n-grams. In: Proceedings of the 2000 workshop on New security paradigms. pp. 101–110. ACM (2001)
30. McCallum, A., Nigam, K., et al.: A comparison of event models for naive bayes text classification. In: AAAI-98 workshop on learning for text categorization. vol. 752, pp. 41–48. Citeseer (1998)
31. Pugelj, M., Džeroski, S.: Predicting structured outputs k-nearest neighbours method. In: International Conference on Discovery Science. pp. 262–276. Springer (2011)
32. Sanchez-Pi, N., Martí, L., Garcia, A.C.B.: Text classification techniques in oil industry applications. In: International Joint Conference SOCO'13-CISIS'13-ICEUTE'13. pp. 211–220. Springer (2014)
33. Santos, I., Nieves, J., Bringas, P.G.: Semi-supervised learning for unknown malware detection. In: International Symposium on Distributed Computing and Artificial Intelligence. pp. 415–422. Springer (2011)
34. Silla Jr, C.N., Freitas, A.A.: A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery 22(1-2), 31–72 (2011)
35. Solovyev, V.V., Makarova, K.S.: A novel method of protein sequence classification based on oligopeptide frequency analysis and its application to search for functional sites and to domain localization. Computer applications in the biosciences: CABIOS 9(1), 17–24 (1993)
36. Sun, A., Lim, E.P., Ng, W.K.: Performance measurement framework for hierarchical text classification. Journal of the American Society for Information Science and Technology 54(11), 1014–1028 (2003)
37. Tan, S.: An effective refinement strategy for knn text classifier. Expert Systems with Applications 30(2), 290–298 (2006)
38. Tomović, A., Janičić, P.: A variant of n-gram based language classification. In: Congress of the Italian Association for Artificial Intelligence. pp. 410–421. Springer (2007)

39. Tripathi, N., Oakes, M., Wermter, S.: Hybrid classifiers based on semantic data subspaces for two-level text categorization. International Journal of Hybrid Intelligent Systems 10(1), 33–41 (2013)
40. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In: Proceedings of the twenty-first international conference on Machine learning. p. 104. ACM (2004)
41. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research 6(Sep), 1453–1484 (2005)
42. Vogrinčič, S., Bosnić, Z.: Ontology-based multi-label classification of economic articles. Computer Science and Information Systems 8(1), 101–119 (2011)
43. Wiśniewski, J.L.: Effective text compression with simultaneous digram and trigram encoding. Journal of Information Science 13(3), 159–164 (1987)
44. Zamora, E., Pollock, J.J., Zamora, A.: The use of trigram analysis for spelling error detection. Information Processing & Management 17(6), 305–316 (1981)

Dr Jelena Graovac is an assistant professor at Faculty of Mathematics, University of Belgrade, Department for Computer Science. Her research interests include natural language processing, information retrieval and document classification using machine learning and knowledge-based approaches.

Dr Jovana Kovacevic is an assistant professor at Faculty of Mathematics, University of Belgrade, Department for Computer Science. She is a member of Bioinformatics research group at the same Faculty and also cooperates with another bioinformatics research group from Indiana University in USA. Her research interests include bioinformatics, data mining and text mining with special emphasis on structural classification in all areas.

Dr Gordana Pavlovic-Lazetic is a full professor at the Faculty of Mathematcics, University of Belgrade. Her research interests include natural language processing, data and text mining, databases and bioinformatics. As a visiting scholar she spent two years at the University of California, Berkeley (USA) working on development of the RDBMS INGRES. She supervised eight Ph.D theses, participated in many national and international projects and conferences, published a large number of scientific papers with many citations.

Received: October 17, 2015; Accepted: October 1, 2016.