Application of a Structural Support Vector Machine Method to N-gram Based Text Classification in Serbian

UDC 811.163.41'322.2
DOI 10.18485/infotheca.2016.16.1_2.1

ABSTRACT: The paper presents classification results that were obtained using the Support Vector Machine method (SVM) over a hierarchically organized corpus of documents in Serbian. Two techniques derived from the SVM with structural output have been applied: multiclass flat classification and hierarchical classification. A common representation model of a document and a class or a hierarchy of classes the document belongs to, specific for this form of the SVM method, is based on different length byte n-grams. Four tf-idf statistics have been used that define significance of an n-gram for a specific document. The described techniques and statistics have been tested on a hierarchically structured subset of the Ebart corpus of newspaper texts. The results obtained for both types of classifiers are similar for the corpus as a whole, while hierarchical classifier performs better for most specific classes with a small number of texts.

KEYWORDS: hierarchical text classification, Support Vector Machine Method, Ebart corpus

PAPER SUBMITTED: 22 May 2015
PAPER ACCEPTED: 30 October 2015

Jovana Kovačević
jovana@matf.bg.ac.rs

Jelena Graovac
jgraovac@matf.bg.ac.rs

University of Belgrade
Faculty of Mathematics
Department for Computer Science and Informatics

1 Introduction

Text classification is one of the text mining tasks – an area of computational linguistics involving a set of techniques for extracting useful, hidden, previously unknown information from textual documents. In the case of text classification, hidden information is a class or a set of classes from a predefined set of classes that a text belongs to, based on its content (Manning and Schütze, 1999). Classification may

be performed manually, but such a process is time consuming and expensive. Availability of high speed computers made automatic classification the basis for efficient processing of large document collections and discovery of knowledge contained in them.

In automatic or semi-automatic text classification two different approaches are mainly used: lexical-semantic language resources based approach and machine learning based approach. Systems of the first type use lexical-semantic networks, such as WordNet (Miller, 1995) and FrameNet (Johnson et al.), along with resources and tools such as electronic dictionaries and lexicon grammars (Gross, 1997), semantic ontologies, named entity and proper names ontologies. These language resources provide for development of a classification model for a morphologically and derivationally exceptionally rich language as Serbian. Classifier is usually built manually based on rich morphological, syntactic and semantic information contained in lexical resources (Scott and Matwin, 1998) and it does not require existence of a set of pre-classified texts to be used for classifier training. As opposed to these, systems that follow another approach assume existence of pre-classified text corpora divided into training and test sets. Based on training data, classifiers (classification models), are built by using different statistical methods, e.g., Bayes classification (Eyheramendy et al., 2003), Conditional Random Fields (McCallum and Pereira, 2001), Hidden Markov Models (Yi and Beheshti, 2013), or methods based on Support Vector Machine model, neural networks (Sebastiani, 2002), nearest neighbors (Yang and Pedersen, 1997), decision trees (Quinlan, 1996), etc. Especially important is the multilingual EuroVoc classifier JEX (Steinberger et al., 2013), comprising of trained classifiers for 22 different European Union languages.

Choice of a method, as well as of an approach to a text classification problem, depends on two key factors: availability of language resources and availability of training data. If language resources such as lexicons, dictionaries and grammars and semantic networks exist, it makes sense to use the resources-based approach. This approach has to take into account characteristics and specificities of each language to be applied to, so in case of Serbian the fact that it uses two alphabets (Cyrillic and Latin), that orthography is phonologically based, morphological system rich, free word order in a sentence, agreement system very complex (Vitas et al., 2003). All these characteristics make preprocessing steps, such as feature selection and feature extraction, based on which classification is performed, quite complex. If corpora needed for training algorithms – in case of text classification those are classified text bases – exist or are easy and inexpensive to develop, it is convenient to apply the machine learning approach. In machine learning techniques, a classifier is generated automatically, by "learning" characteristics of classes based on a training dataset associated to each class. Data in these datasets are manually classified by domain

experts. After the training process, the classifier usually automatically generates a set of rules the data item has to satisfy in order to be classified in a specific class.

Depending on the number of classes, classification may be binary, when only two classes are defined or multiclass, when more than two possible classes are defined. Depending on whether classes may overlap or not, classification may be single-label, when a data item may be associated exactly one class, or multi-label, when a data item may be assigned one, zero or more than one class, i.e., classes may overlap. According to the structure defining relationship between classes, classification may be hierarchical or non-hierarchical. If classes are treated independently, without any structure defining relationships between them, it is a non-hierarchical classification. When the number of different classes, or the number of data items inside one class becomes very large, problems arise with accurate and efficient searching and managing of data at a class level. In that case classes are usually organized into tree-like structures and a hierarchical structure is introduced among them (Sun and Lim, 2001) (e.g., Yahoo hierarchy).

In (Graovac, 2014b) and (Pavlović-Lažetić and Graovac, 2010) a document classification method for Serbian is presented based on Serbian WordNet (Krstev et al., 2004) developed for Serbian by the Language Technology Group at the University of Belgrade, Faculty of Mathematics[1] and applied to the newspaper texts corpus Ebart[2]. Other language resources developed for Serbian by the Group have also been used – electronic dictionary (Vitas and Krstev, 2005), lexicon grammars (Vitas et al., 2003), proper names ontologies (Krstev et al., 2005).

In (Graovac et al., 2015; Graovac, 2014a; Graovac and Pavlović-Lažetić, 2014; Graovac, 2014b) machine learning – based classification methods are presented using n-gram method for text representation and k nearest neighbor method (kNN) for classifier development. Methods are language independent, applied to text corpora of the most widespread writings and languages, with different lexical, morphological, syntactic and orthographic characteristics (English, Chinese, Arabic, and Spanish), they are very simple and performed very well. In (Graovac, 2012), application of the machine learning n-gram method was presented to text classification in Serbian on the newspaper texts corpus Ebart. Classification is multiclass, multi-label and non-hierarchical. In this paper the structural support vector machine method is applied for the first time to text classification in Serbian (Ebart corpus). A hierarchical class structure has been defined over the flat corpus, and a sub-corpus has been extracted from the original corpus, consisting of documents corresponding, by content, to the classes in the hierarchical structure. Then two classification methods are applied to the hierarchically organized corpus, both inferred from the SVM with structured output (SSVM): multiclass (flat) classification (selection of one out of many classes

[1] www.matf.bg.ac.rs/∼cvetana/LT-pregled.html
[2] http://www.arhiv.rs/novinska-arhiva/

and hierarchical classification (selection of a class hierarchy the document belongs to). The common representation model of a document and a class or a class hierarchy the document belongs to, specific for this form of the SVM method, is based on different length byte n-grams.

Last, the model has been evaluated based on test data and accuracy of results has been determined by using one of the standard information retrieval measures – F1 measure combining recall and precision (Tan et al., 2006).

In what follows, the Ebart corpus will be presented first – the largest digital media documentation in Serbia, as well as the hierarchically organized sub-corpus extracted from the Ebart corpus for the purpose of testing the hierarchical classification methods (part 2).

Then the applied methodology will be sketched (part 3): SVM method with structured output and its adjustments for application to multiclass and hierarchical classification (section 3.1), byte n-gram concept (section 3.2) and a specific n-gram common representation of document and class (or class hierarchy) the document belongs to, as well as training and testing steps in application of this method (section 3.3). Evaluation measures are introduced in the section 3.4.

The main result of the paper – text classification result – will be presented in part 4. Evaluation results will also be presented as well as comparison with related (comparable) results. Finally, in part 5, we shall interpret and discuss obtained results, impact of applied methods and improvement possibilities.

2 Dataset

Ebart corpus represents the largest digital documentation of newspaper texts in contemporary Serbian language. It has been created in 2003 and up to nowadays it has warehoused more than 2,000,000 texts from the printed media. Last version of Ebart archive is classified according to theme units resembling the common columns in the newspapers: politics, foreign affairs, society, economy, chronicle, culture, fun, sports, media, feuilleton, readers' letters, and so on. In order to test structural support vector machines method on the problem of hierarchical text classification, we extracted a hierarchically organized sub-corpus of "flat" Ebart corpus and named it EbartHier. EbartHier includes all the articles from the daily newspaper "Politika", published in from 2003 to 2006, that belong to the following columns: Politics, Society, Economy and business, World economy and finances, Culture, Science and Technology, as well as all the articles from the "Sportski zurnal" magazine (published from 2003 to 2006) that belong to columns Basketball and Football. All the documents that belong to columns/classes that are similar by topic, have been grouped in the same class on the higher hierarchy level. In this way, classes Politics and society, Economy, Culture and science and Sport have been created. The obtained corpus

has a treelike structure represented in Figure 1. Classes are non-overlapping (one document can belong only to one class) and each document can be classified only in a class that is situated in the leaf of the hierarchy. Corpus is characterized by extremely uneven distribution of documents by classes (see Figure 1). The median of lengths of all the documents from the corpus is 327.75 words in a document (the shortest one has 7 and the longest one has 2576 words). Figure 2 represents a histogram of medians of documents' lengths (including the length of the shortest and the longest document) by all classes. All documents in the corpus are represented in Serbian Latin alphabet using the UTF-8 code scheme.

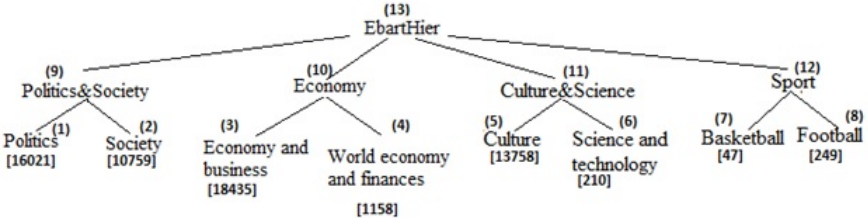

Figure 1. Treelike structure of EbartHier corpus. Each class name is assigned a number corresponding to that class. In the leaves, in square brackets, number of documents per class is displayed

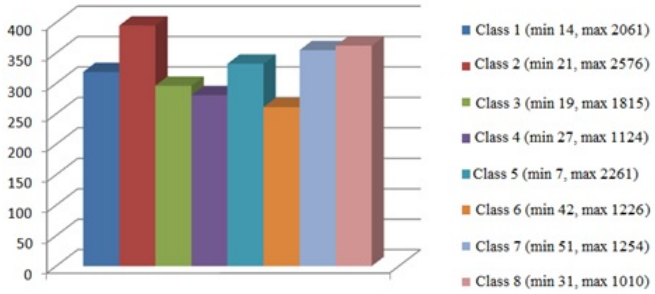

Figure 2. Histogram of mean lengths of documents in classes. Numbers in brackets correspond to the lengths of the shortest and the longest document in each class.

3 Methodology

3.1 Structural support vector machines method

Support Vector Machines method (SVM) has shown to be very efficient in text classification (Joachims, 1998). In this paper, we used Structural Support Vector Machines method (SSVM, (Tsochantaridis et al., 2004)) which represents generalization of the SVM method on structural output, for example array, tree, directed acyclic graph etc.

Before we represent the SSVM method, we will give a brief overview of the SVM method from the perspective of text classification. A standard approach in training predictors for binary classification is to learn the discriminant function $F(x)$ and to classify the input vector x according to the sign of the function $F(x)$. Since linear methods usually have efficient learning algorithms, it is common to assume that the function $F(x)$ is linear. Following this assumption, we can represent function $F(x)$ in the form $F(x) = \langle \omega, x \rangle$, where ω is the vector of learning parameters and „$\langle \dots \rangle$" is the notation for scalar product. Input vector x can be mapped into another space using the function Ψ and in that case, we write function $F(x)$ in the following way: $F(x) = \langle \omega, \Psi(x) \rangle$.

Binary classifier can predict whether a document belongs to a certain class or not, which means that its output can be -1, if the document does not belong to that class, or 1 otherwise. If we want to answer the question "Which class does the document belong to?" we need to turn to methods that predict structural output, namely SSVM. In the basic SVM method, output y is determined according to the sign of the discriminant function, that is $sgn(F(x)) = y$, where $y \in \{-1, 1\}$. It would be ideal if we could find analogous discriminant function $F(x)$ that maps the input dataset (in this case, corpus of documents) into the output dataset (in this case, classes). Since it is very difficult to create such function, we turn to the next best solution and create a function $F : X \times Y \to \mathbb{R}$ that measures how well the output y corresponds to the input x.

We would like to create the function F such that the larger value of $F(x, y)$, the better the output y corresponds to the input x, that is, in this specific case, given class better corresponds to the given text. In this generalization, the discriminant function becomes the function of two arguments, input and output, $F(x, y)$ where the output is not from the set $\{-1, 1\}$ yet it can represent array, tree, graph, etc. If we denote the set of all possible outputs by Y, no matter whether it contains arrays, graphs, trees or some other structure, SSVM predicts the output (class) that corresponds best to the input (text), i.e. it predicts the output vector y that maximizes the value of the function F for a given input vector x. To be more precise, SSVM predicts output based on the following equation: $y^* = \underset{y \in Y}{\operatorname{argmax}}(F(x^*, y))$. Analogously with

SVM, in SSVM we also assume that function F is linear in ω, as well as that the pair of vectors (x, y) can be more suitably represented by mapping into some other space using a function Ψ. Therefore, we can write function F as $F(x, y) = \langle \omega, \Psi(x, y) \rangle$.

Function Ψ represents a joint input-output vector for one input-output pair (x, y) and its form depends on the dataset that the method is being applied to. For example, one of applications of SSVM is creating ("predicting") a derivation tree in the given formal grammar and in that case, input vector x would represent a vector of words that appear in a sentence, output vector y would represent a derivation tree in the formal grammar and function $\Psi(x, y)$ would be a joint representation of the sentence and its derivation tree. This joint representation could be a vector whose dimension is equal to the total number of derivation rules, including rules of derivation of all words from the training dataset. Each element of this vector would correspond to one of all possible rules of the grammar, and its value would be equal to the total number of occurrences of the that rule. This representation is illustrated by the example in Figure 3.

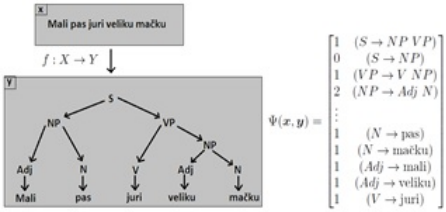

Figure 3. Example of joint representation of input vector x and output vector y, constructed according to the similar example of derivation in English grammar in (Tsochantaridis et al., 2004)

In the example shown in Figure 3, vector x consists of the words from the given sentence ("Mali pas juri veliku mačku", eng. "A small dog is chasing a big cat"), and vector y represents a derivation tree in the given formal grammar. Vector $\Psi(x, y)$ denotes that, in the sentence derivation, rule $S \to NPVP$ has been applied once, rule $S \to NP$ zero times, rule $VP \to VNP$ once, rule $NP \to AdjN$ two times and so on, rule $V \to juri$ once.

There are different formulations of the SSVM method (Tsochantaridis et al., 2005). In this paper, we used the so-called 1-slack margin rescaling formulation (Joachims et al., 2009). When a classifier is being trained, it is good for the margin

that separates the training examples to be as wide as possible, whereas on the other hand, wide margin can cause misclassification of some training examples. The 1-slack margin rescaling formulation learns parameters ω depending on a positive constant C that controls the trade-off between minimizing training error and maximizing the margin (this constant will be especially analyzed further in the paper). The described algorithm has polynomial complexity by the number of training examples, which has been proven in (Joachims et al., 2009).

3.2 N-grams

Let s be a string of symbols $s = s_1 s_2 ... s_N$ over alphabet Σ. An n-gram of the string s (for natural numbers n and N) is defined as any substring of adjacent symbols of the string s of length n. Totally $|\Sigma|^n$ different n-grams can be defined over the alphabet Σ, where $|\Sigma|$ is its size (cardinality). N-grams represented in this way can be defined on word level, character level or byte level. For example, 2-grams (usually referred to as bigrams) over the string "dela, ne reči" on word level will contain only two bigrams "dela ne" and "ne reči", whereas character level bigrams (alphabet Σ is Serbian Latin) will be de; el; la; a,; ,_; _n; ne; e_; _r; re; eč; či. If the characters are coded by UTF-8 code scheme, letter "č" is coded by two bytes whose decade content is 196 141, respectively, character " " (space) is coded by one byte whose content is decade number 32 and so on. In that way, the entire string is written in the computer by an array of bytes with decade values 100 101 108 97 44 32 110 101 32 114 101 196 141 105. Thus, in case of languages over the Latin alphabet, n-grams on byte level and n-grams on character level are quite similar considering the fact that one character is mostly represented by one byte. The difference is usually also in the set of characters that is being considered (n-grams on the character level usually do not take into account the difference between big and small letters, punctuation symbols and digits), and the difference is especially significant when Cyrillic alphabet is used, or other alphabets like Arabic, Chinese etc. N-grams of bytes and n-grams of characters are equally used for text representation in solving different data mining tasks (Kešelj et al., 2003; Abou-Assaleh et al., 2004; Reddy and Pujari, 2006; Santos et al., 2011; Lui et al., 2014), 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 code scheme, which is why they represent simplified representation for computer processing. In this paper, we will use byte n-grams.

When used in natural language processing, some of the advantages of n-grams include relative insensitivity to spelling mistakes, the alphabet of symbols is known

in advance, independency of language and content, execution in one run, no need for any previous linguistic knowledge, and so on.

The basic problem with using n-grams is their exponential number with respect to the alphabet's cardinality. If Σ is English language alphabet and if we attach a space symbol to it, than $|\Sigma| = 27$. If we make a difference between capital and small letters and if we also include digits, than $|\Sigma| = 63$. It is clear that many algorithms with n-grams will be very expensive from the computational point of view even when $n = 5$ or $n = 6$ (for example, number of different 5-grams over the alphabet Σ is $63^5 \sim 10^9$).

Using models and techniques based on n-grams for natural language processing has shown to be an efficient approach. This approach has application in information retrieval (De Heer, 1974), text compression (Wiśniewski, 1987), detecting and correction of grammar mistakes (Zamora et al., 1981), detecting document's authorship (Kešelj et al., 2003) and so on.

3.3 Data representation

During this research, we developed two classifiers based on the original SSVM algorithm which have different representations of output data, in this case classes of documents. Both classifiers use the same representation of input data, namely byte level n-grams. Each position in n-gram vector is equal to tf-idf statistics' value for the given n-gram. Tf-idf statistics (tf-idf is short from term frequency-inverse document frequency) is usually defined in a way to reflect importance of an n-gram for a document in a corpus. This measure is directly proportional to the number of occurrences of the n-gram in the document, but it is also inversely proportional to the number of occurrences of the n-gram in the entire corpus. N-grams that have higher value of tf-idf statistics will be more significant for the document, more precisely, n-grams that occur more frequently in the document but are not that frequent in the entire corpus.

There are different variations for calculating values of tf and idf measures. In this paper, we used the following (Manning et al., 2008):

1. *classic tf-idf*: $tf \cdot \log\left(\frac{n}{n_k+1}\right)$
2. *log tf-idf*: $1 + \log(tf) \cdot \log\left(\frac{n}{n_k+1}\right)$
3. *boolean1 tf-idf*: $\log\left(\frac{n}{n_k+1}\right)$
4. *boolean2 tf-idf*: $\log\left(1 + \frac{n}{n_k}\right)$

where tf represents normalized frequency of n-grams in the corresponding document, n represents total number of documents in the entire corpus, and n_k represents number of documents in the corpus in which that n-gram occurs at least once.

Representation of output data, in this case classes, is different for two developed classifiers. Each document of EbartHier corpus is assigned one class situated in the leaf of the EbartHier hierarchy.

In the first classifier, each class is represented as a unique natural number, without considering the hierarchical relationship of the classes. If we enumerate the classes respectively as in Figure 1, set Y becomes equivalent to the set $\{1, 2, 3, 4, 5, 6, 7, 8\}$. In this way, we adjusted the basic SSVM method to work as a multiclass flat classifier.

In the first classifier, vector $\Psi(x, y)$, joint representation of input and output vectors, has dimension of $p \cdot q$, where p is the number of different n-grams, i.e. dimension of the vector x, and q is the number of different classes in the corpus. In that way, each class gets its block of size p in vector Ψ, which will contain zeros if the given document does not belong to the given class or values of the input vector x, otherwise. For example, if a document x belongs to class k, then the joint representation will be the following:

$$\Psi(x, y) = \left[\underbrace{0, \ldots, 0}_{\text{class 1}}, \ldots, \underbrace{x_1, \ldots, x_p}_{\text{class k}}, \ldots, \underbrace{0, \ldots, 0}_{\text{class q}} \right]$$

In the second classifier, each class is represented as a vector of nodes in EbartHier hierarchy. Each position in vector y corresponds to one node in the hierarchy, with value "1" if the node occurs in the path from the root to class' leaf and „0" otherwise. In the example in Figure 1, class economy and business would be represented as $(0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1)$. In this way, we adjusted the basic SSVM method to work as a hierarchical classifier.

In the second classifier, vector $\Psi(x, y)$ has dimension of $p \cdot r$, where p is the number of different n-grams, i.e. dimension of the vector x, and r is the number of nodes in the hierarchy. In that way, each node gets its block of size p in vector Ψ, which will contain zeros if the given document does not belong to the class that contains the given node or values of the input vector x, otherwise. For example, if document x belongs to class economy and business, represented as $(0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1)$, then the joint representation will be the following: $\Psi(x, y) = [0, 0, x, 0, 0, 0, 0, 0, 0, x, 0, 0, x]$ where 0 is the zero vector $0 = [\underbrace{0, \ldots, 0}_{\text{p times}}]$, and vector x corresponds to the input vector.

The basic difference in representations of vector y in flat and hierarchical classifier is that the flat one treats all classes individually, whereas the hierarchical one takes into account that the classes are part of a treelike hierarchy. This difference is reflected in the joint representation of input and output vectors, vector $\Psi(x, y)$, which has different dimensions in these two cases: in flat classifier, each position in vector Ψ is reserved for one class (which is situated in the hierarchy leaves) whereas

in hierarchical classifier each position is reserved for one node of the hierarchy, including leaves, inner nodes and root.

3.4 Implementation

Both classifiers have been constructed by adjusting publicly available and free SVM$^{\text{struct}}$[3] framework for SSVM method. SVM$^{\text{struct}}$ is available in several programming languages and for this purpose we used its implementation in programming language C. Adjustment of the basic implementation included adapting existing data structures for input vector x and output vector y, implementation of function which generates vector $\Psi(x, y)$ (joint representation of input vector x and output vector y), implementation of the loss function and implementation of the function for evaluating classifiers' quality.

The data have been prepared in the following way:

1. For each document, Text::Ngrams (Kešelj et al., 2003) tool has been used to generate the array of byte n-grams that the document contains as well as how many times each byte n-gram appears. Data for byte n-grams of length $\{2, 3, 4, 5, 6, 7\}$ have been generated.
2. We implemented a script that for a given document, based in its n-grams, generates input vector x in the specific format required by the SVM$^{\text{struct}}$ classifier

3.5 Experiments

We divided the EbartHier corpus of documents to training set and test set in proportion 2:1, which is one of the most frequent ways of division in text classification (Bellotti and Crook, 2009). From the entire corpus of 60,637 documents, the training set contains 40,426 documents and the test set 20,211 documents. Distribution of documents by classes also follows this proportion. It is displayed in Table 1.

Each document of the EbartHier corpus is represented by byte n-grams which determine the vector representation of the document suitable for the classifier, the so-called input vector x. Each byte n-gram from the corpus is assigned one position in the vector x and the value on the position is equal to the value of one of 4 tf-idf statistics for the corresponding byte n-gram. Depending on the length of byte n-gram ($n \in \{2 \ldots 7\}$) and on the chosen tf-idf measure (classic, log, boolean1, boolean2), we generated 24 representations of the corpus ($n = 2$, measure=classic, ..., $n = 7$, measure=classic, $n = 2$, measure=log, ..., $n = 7$, measure=log, ..., $n = 2$, measure=boolean1, ..., $n = 7$, measure=boolean1, $n = 2$, measure=boolean2, ..., $n = 7$, measure=boolean2). In each representation, training sets and test sets consisted

[3] http://www.cs.cornell.edu/people/tj/svm_light/svm_struct.html

of the same documents so that the performance of classifiers could be compared on them.

In order to investigate whether a certain type of classifiers gives better results for certain length of byte n-grams or for certain tf-idf statistics, we performed flat and hierarchical classification for each corpus representation in the following way:

1. In order to find the optimal value of parameter C, we performed 10-cross validation on each training set.
2. For the value obtained, we have trained the classification model on the entire training set.
3. Testing and model evaluation have been performed on the test set.

At the end, we compared the performance of the trained classification models.

Class	Traning set	Test set	Entire corpus
1	10681	5340	16021
2	7173	3586	10759
3	12290	6145	18435
4	772	386	1158
5	9172	4586	13758
6	140	70	210
7	32	15	47
8	166	83	249
Total	40426	20211	60637

Table 1. Number of documents in the training set and in the test set by classes

3.6 Evaluation

In order to analyze performances of SSVM methods of hierarchical classification, we use the usual measures for quality of classification – precision (the percentage of correctly classified examples from all the examples assigned to a certain class), recall (number of test examples of the certain class that the classifier can recognize) and F-measure (F1), harmonic mean of precision and recall (Baeza-Yates et al., 1999):

$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

where *TP* (True Positives) is the number of correctly positively classified documents, *TN* (True Negative) is the number of correctly negatively classified documents, *FP* (False Positive) is the number of incorrectly positively classified documents and *FN* (False Negative) is the number of incorrectly negatively classified documents.

These measures are defined for the case of binary classification (when there are only two classes). In case when there are more than two classes, it is necessary to find mean value of these measures by all classes. That can be achieved in two ways: by calculating macro-average, where each class is equally significant, or by calculating micro-average, where classes with larger number of documents are being favored. In macro-average, we first calculate measures for each class individually and then find mean value by the number of classes. In micro-average, we calculate values for *TP, TN, FP* and *FN* for each class individually and then we calculate values *TP, TN, FP* and *FN* by summing all *TP, TN, FP* and *FN* for all classes respectively. At the end, we calculate measures for the sums *TP, TN, FP* and *FN*. In this paper, we will use micro-average of F-measure (micro-F1). Basic disadvantage of these measures is that errors made on different levels of hierarchy are equally punished.

4 Results

We applied two variants of the SSVM method on the EbartHier corpus: in the first one, we performed flat classification, not taking into account hierarchical relationship of the classes, and in the second one we performed hierarchical classification. For every document from the corpus, 6 different n-gram representations have been generated, for byte n-grams of length from 2 to 7, for 4 different tf-idf measures presented in section 3.3. On each training set obtained in this way, we performed 10-cross validation which determined optimal value of the classifiers' parameter C, from the set of values from 10^{-2} до 10^{2}, with step 10. Figure 4 shows evaluation results for both types of classifiers, represented by micro-F1 measure, for corpus represented with byte n-grams of length from 2 to 7 and for all 4 tf-idf measures whereas Table 2 shows complete numerical data. Both types of classifiers (flat and hierarchical) show the same tendency that classifiers with classic measure have worse performances than others. Values of micro-F1 for other measures are approximate. Differences between two micro-F1 measures for different classifiers (flat and hierarchical) with the same measure are almost always under 1% (except for measure classic, where the differences are 2.32% and 1.66% for $n = 2$ and $n = 3$). Performances of the best classifiers for both types (flat and hierarchical) have been analyzed by classes. Among flat classifiers, classifier with measure boolean1 and byte n-gram length of 6 had the highest value of micro-F1 measure (90.43%) and among hierarchical classifiers, the

most successful was the classifier with measure boolean2 and byte n-gram length of 4 (micro-F1 is 90.17%). Considering that the number of documents by classes is different (displayed in Figure 1), classes can be divided into two groups: "large" (1, 2, 3 and 5) and "small" (4, 6, 7 and 8). Figure 5 shows the results of the analysis described. Results for two best classifiers are approximate for the "large" classes whereas differences are more distinguished for the "small" classes. Best hierarchical classifier is much better in predicting classes 6 and 7 than the best flat classifier. Both classifiers show highest accuracy for "small" class denoted by 8 (football).

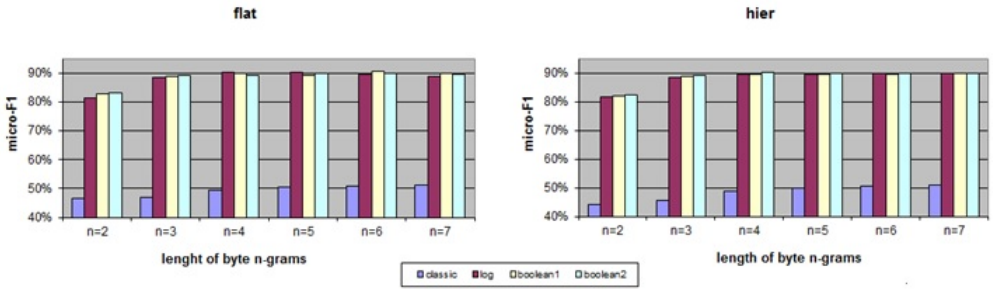

Figure 4. Results of the evaluation for two types of classifiers (flat and hierarchical) for different input data (byte n-gram's length from 2 to 7) and different measures

flat	classic	log	boolean1	boolean2
n=2	46.53%	81.46%	82.71%	83.25%
n=3	47.08%	88.27%	88.70%	89.28%
n=4	49.33%	90.05%	90.02%	89.18%
n=5	50.35%	90.05%	89.23%	89.77%
n=6	50.89%	89.60%	90.43%	89.74%
n=7	51.12%	88.88%	89.88%	89.60%

hier	classic	log	boolean1	boolean2
n=2	44.21%	81.75%	82.04%	82.45%
n=3	45.42%	88.34%	88.86%	89.24%
n=4	48.61%	89.39%	89.58%	90.17%
n=5	49.84%	89.46%	89.57%	89.70%
n=6	50.65%	89.84%	89.47%	89.78%
n=7	51.01%	89.88%	89.68%	89.69%

Table 2. Performances of flat (left table) and hierarchical (right table) classifier for different representations in input corpus, expressed by micro-F1 measure

5 Conclusion

Text classification of documents in Serbian language from a hierarchically organized corpus using the SSVM method shows approximate results for flat and hierarchical variants, for every tf-idf measure.

In total, the best result was achieved by the flat variant of the classifier for the boolean1 measure and its micro-F1 measure is 90.43%. This is a slightly better result than the previously published result for n-gram classification of a flat subset of the Ebart corpus where the micro-F1 measure was 88.5% (Graovac, 2012). Hierarchical classification model gives slightly better results than the flat one for some small classes.

Hierarchical classification presents weaker results than expected, which can be explained by the shallow hierarchy with a small number of documents, but also by the evaluation measure used. In evaluating classification results, especially hierarchical, it is not enough to count incorrectly classified examples, but it is also necessary to estimate their weight, i.e. the distance of the predicted class from the true class.

Having that in mind, we expect that by using better measure of evaluation, adapted for hierarchical classification, we could achieve results that outperform the results of the flat classification (Silla et al., 2011).

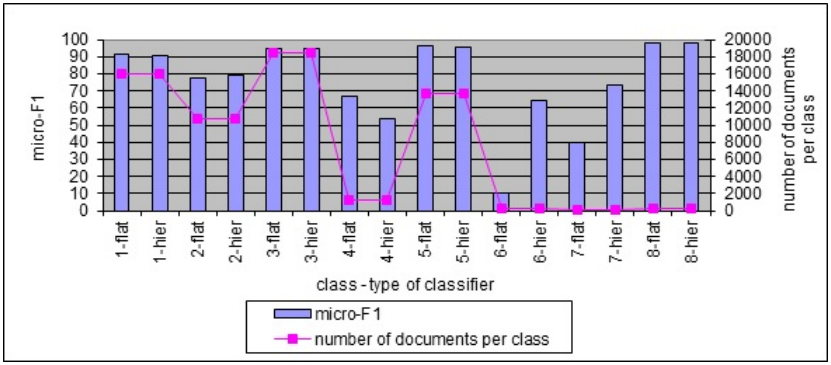

Figure 5. Results of the evaluation for the best classifiers for both types (flat and hierarchical). Number of documents in corpus per each class is also displayed.

References

Abou-Assaleh, Tony, Nick Cercone, Vlado Kešelj, and Ray Sweidan. "N-gram-based detection of new malicious code". In *Computer Software and Applications Conference*, COMPSAC 2004. Proceedings of the 28th Annual International, IEEE. Vol. 2 (2004), 41-42. Accessed September 1, 2015. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1342667.

Baeza-Yates, Ricardo, Ribeiro-Neto Berthier, et al. *Modern information retrieval*, vol. 463. ACM press New York, 1999.

Bellotti, Tony and Jonathan Crook. "Support vector machines for creditscoring and discovery of significant features". *Expert Systems with Applications*, vol. 36, no. 2 (2009): 3302-3308. Accessed September 1, 2015. http://www.sciencedirect.com/science/article/pii/S0957417408000857.

De Heer, T. "Experiments with syntactic traces in information retrieval". *Information Storage and Retrieval*, vol. 10, no. 3 (1974): 133–144. Accessed September 1, 2015. http://www.sciencedirect.com/science/article/pii/0020027174900151.

Eyheramendy, Susana, David D. Lewis, and David Madigan. "On the Naive Bayes model for text categorization". In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics Conference*, 332-339. Florida: Society for Artificial Intelligence and Statistics, 2003.

Graovac, Jelena. "Serbian text categorization using byte level n-grams". In *Proceedings of CLoBL 2012: Workshop on Computational Linguistics and Natural Language, 5th Balkan Conference in Informatics*, 93–97. 2012 Accessed September 1, 2015. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.416.6155&rep=rep1&type=pdf.

Graovac, Jelena. "Wordnet-based text categorization technique". *INFOtheca – Journal of Information and Library Science*, vol. 14, no. 2 (2013): 2–17. Accessed September 1, 2015. http://infoteka.unilib.rs/2013/br.2/eng/Infotheca-2-2013-Jelena-Graovac.pdf.

Graovac, Jelena. "A variant of n-gram based language-independent text categorization". *Intelligent Data Analysis*, vol. 18, no. 4 (2014): 677–695.

Graovac, Jelena. "Text categorization using n-gram based language independent technique". Natural Language Processing for Serbian - Resources and Applications. In *Proceedings of the Conference "35th Anniversary of Computational Linguistics in Serbia"*, 124–135, 2014. Accessed September 1, 2015. http://poincare.matf.bg.ac.rs/~jgraovac/publications/jg35CLS.pdf.

Graovac, Jelena and Gordana Pavlović-Lažetić. "Language-Independent Sentiment Polarity Detection in Movie Reviews: A Case Study of English and Spanish". In *6th International Conference ICT Innovations 2014*, 13–22, 2014. Accessed September

1, 2015. http://poincare.matf.bg.ac.rs/~jgraovac/publications/jgSPD.pdf.

Graovac, Jelena and Jelena Kovačević, and Gordana Pavlović-Lažetić. "Language Independent n-Gram-Based Text Categorization with Weighting Factors: A Case Study". *JIDM - Journal of Information and Data Management*, vol. 6, no. 1 (2015): 4–17.

Gross, Maurice. "The construction of local grammars". In *Finite State Language Processing eds. Emmanuel Roche and Yves Schabs*, 329–354. Massachusetts: The MIT Press, 1997. Accessed September 1, 2015. https://halshs.archives-ouvertes.fr/halshs-00278316/document.

Joachims, Thorsten. "Text categorization with support vector machines: Learning with many relevant features". Springer Berlin Heidelberg, 1998.

Joachims, Thorsten, Thomas Finely, and Chun-Nam John Yu. "Cutting Plane Training of Structural SVMs". *Machine Learning Journal*, vol. 77, no. 1 (2009): 27–59. Accessed September 1, 2015. http://link.springer.com/article/10.1007/s10994-009-5108-8.

Johnson, Christopher R., Miriam R. L. Petruck, Collin F. Backer, Michael Ellsworth, Josef Ruppenhofer, and Charles J. Fillmore. "FrameNet: Theory and Practice", 2002. Accessed September 1, 2015. http://www.icsi.berkeley.edu/framenet.

Kešelj, Vlado, Fuchung Peng, Nick Cercone, and Calvin Thomas. "N-gram-based author profiles for authorship attribution". In *Proceedings of the conference pacific association for computational linguistics, PACLING*, vol. 3, 255–264. 2003. Accessed September 1, 2015. http://web.cs.dal.ca/~vlado/papers/pacling03.pdf.

Krstev, Cvetana, and Gordana Pavlović-Lažetić, and Ivan Obradović. "Using textual and lexical resources in developing Serbian wordnet". *Romanian Journal of Information Science and Technology*, vol. 7, no. 1–2 (2004): 147–161. Accessed September 1, 2015. http://xn--c1azn.xn--90a3ac/LicnePrezentacije/ivan_obradovic/Radovi/RJIS_2004.pdf.

Krstev, Cvetana, Duško Vitas, Denis Maurel, and Mickaël Tran. "Multilingual ontology of proper names". In *Proceedings of 2nd Language & Technology Conference*, April 21-23, 2005, Poznań, Poland, ed. Zygmunt Vetulani, 116–119. Poznań, Wydawnictwo Poznańskie, 2005. Accessed September 1, 2015. https://hal.archives-ouvertes.fr/hal-01108242/document.

Lui, Marco, Jey Han Lau, and Timothy Baldwin. "Automatic detection and language identification of multilingual documents". *Transactions of the Association for Computational Linguistics 2* (2014): 27–40. Accessed September 1, 2015. https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/86/30.

Manning, Christopher, and Hinnrich Schütze. *Foundations of Statistical Natural Language Processing*. Cambridge: MIT Press, 1999.

Manning, Christopher, Prabhakar Raghavan, and Hinnrich Schütze. "Scoring, term weighting, and the vector space model". In *Introduction to Information Retrieval*. Cambridge: Cambridge University Press, 2008.

Lafferty, John, Andrew McCallum and Fernando C. N. Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data". *Machine Learning*, 282–289. 2001

Miller, George A. "WordNet: a lexical database for English". *Communications of the ACM*,vol. 38, no. 11 (1995): 39–41. Accessed September 1, 2015. http://nlp.cs.swarthmore.edu/~richardw/papers/miller1995-wordnet.pdf.

Pavlović-Lažetić, Gordana and Jelena Graovac. "Ontology-driven conceptual document classification". In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*, 383–386. 2010. Accessed September 1, 2015. http://poincare.matf.bg.ac.rs/~jgraovac/publications/odcdc.pdf.

Quinlan, J. R. "Improved use of continuous attributes in c4. 5". *Journal of Artificial Intelligence Research*, vol. 4 (1996): 77–90. Accessed September 1, 2015. http://www.jair.org/media/279/live-279-1538-jair.pdf.

Reddy, D. Krishna Sandeep, and Arun K. Pujari. "N-gram analysis for computer virus detection". *Journal in Computer Virology*, vol. 2, num. 3 (2006): 231–239. Accessed September 1, 2015. http://link.springer.com/article/10.1007/s11416-006-0027-8.

Santos, Igor, Javier Nieves, and Pablo G. Bringas. "Semi-supervised learning for unknown malware detection". In *International Symposium on Distributed Computing and Artificial Intelligence* vol. 91, 415-422. Berlin Heidelberg, 2011.

Scott, Sam, and Stan Matwin. "Text classification using wordnet hypernyms". In *Usage of WordNet in Natural Language Processing Systems: Proceedings of the Workshop*, 38-44, 1998. Accessed September 1, 2015. http://www.aclweb.org/website/old_anthology/W/W98/W98-0706.pdf.

Sebastiani, Fabrizio. "Machine learning in automated text categorization". *In ACM computing surveys (CSUR)*, vol. 34, no. 1 (2002): 1–47.

Silla J., Carlos N. and Alex A. Freitas. "A survey of hierarchical classification across different application domains". *Data Mining and Knowledge Discovery*, vol.22, no. 1–2 (2011): 31–72. Accessed September 1, 2015. http://link.springer.com/article/10.1007/s10618-010-0175-9.

Steinberger, Ralf, Mohamed Ebrahim, and Marco Turchi. "JRC EuroVoc Indexer JEX-A freely available multi-label categorisation tool". *arXiv preprint arXiv*, vol. 1309, no. 5223 (2013). Accessed September 1, 2015. http://arxiv.org/ftp/arxiv/papers/1309/1309.5223.pdf.

Sun, Aixin, and Ee-Peng Lim. "Hierarchical text classification and evaluation". In *Proceedings 2001 IEEE International Conference on Data Mining*, 521–528. Computer Society, 2001.

Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*, vol. 1. Pearson Education India, 2006.

Tsochantaridis, Ioannis, Thomas Hofman, Thorsten Joachims, and Yasemin Altun. "Support Vector Machine Learning for Interdependant and Structured Output Spaces". In *Proceedings of the twenty-first international conference on Machine learning*. 104. New York: ACM, 2004.

Tsochantaridis, Ioannis, Thomas Hofman, Thorsten Joachims, and Yasemin Altun. "Large Margin Methods for Structured and Interdependant Output Variables". *Journal of Machine Learning Research*,vol 6, (2005): 1453–1484. Accessed September 1, 2015. http://machinelearning.wustl.edu/mlpapers/paper_files/TsochantaridisJHA05.pdf.

Vitas, Duško and Cvetana Krstev. "Derivational morphology in an e-dictionary of Serbian". In *Proceedings of 2nd Language & Technology Conference*. April 21-23, 2005, Poznań, Poland, ed. Zygmunt Vetulani, 139-143. Poznań: Wydawnictwo Poznańskie, Accessed September 1, 2015. http://poincare.matf.bg.ac.rs/~cvetana/biblio/ltc_134_vitas_2.pdf.

Vitas, Duško , Cvetana Krstev, Ivan Obradović, Ljubomir Popović, and Gordana Pavlović-Lažetić. "An overview of resources and basic tools for processing of Serbian written texts". In *Proceedings of the Workshop on Balkan Language Resources, 1st Balkan Conference in Informatics, Thessaloniki, Greece*, 97–104, (2003). Accessed September 1, 2015. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.73.8096&rep=rep1&type=pdf.

Wiśniewski, Janusz L. "Effective text compression with simultaneous digram and trigram encoding". *Journal of Information Science*, Vol. 13, No.3, (1987): 159–164.

Yang, Yiming, and Jan O. Pedersen. "A comparative study on feature selection in text categorization". In *Proceedings of the 14th International Conference on Machine Learning*, 412–420, 1997. Accessed September 1, 2015. http://www.surdeanu.info/mihai/teaching/ista555-spring15/readings/yang97comparative.pdf.

Yi, Kwan, and Jamshid Beheshti. "A text categorization model based on Hidden Markov models". In *Proceedings of the 31st Annual Conference of the Canadian Association for Information Science*, 275–287, 2013. Accessed September 1, 2015. http://www.cais-acsi.ca/ojs/index.php/cais/article/view/420/585.

Zamora, E. M., Joseph J. Pollock, and Antonio Zamora. "The use of trigram analysis for spelling error detection". *Information Processing & Management*,vol. 17, no. 6 (1981): 305–316. Accessed September 1, 2015. http://www.sciencedirect.com/science/article/pii/0306457381900443.