Преглед НЦД 29 (2016), 42–51

Jovana Kovačević, Jelena Graovac
Faculty of Mathematics, University of Belgrade, Serbia

PROSPECTIVE AUTOMATED HIERARCHICAL CLASSIFICATION OF DIGITIZED DOCUMENTS

Abstract. The paper presents a proposal of a method for hierarchical classification of digitized documents of NCD digital library. The classification model implements Structured Support Vector Machine method (*SSVM*) which has shown excellent performance on Ebart corpus of documents in Serbian language. We describe the developed model and its results on Ebart dataset, suggest two types of hierarchies of classes of the NCD library regarding its content and define a protocol for the application of the method to digitized documents.

Keywords. Hierarchical text classification, Structured Support Vector Machine Method, n-grams, Ebart

Introduction

Text classification represents one task in the area of text mining. It is a part of the field of computational linguistics that involves a set of techniques for extracting useful, hidden, previously unknown information from textual documents. In 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 [1]. 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.

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 assigned 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 nonhierarchical classification. When the number of different classes, or the number of data items inside one class becomes very large, we are faced with problems regarding accurate and efficient searching and managing data at a class level. In that case, classes are usually organized into tree-like structures and a hierarchical structure is introduced among them [2].

In this paper, we present one method for hierarchical classification and its performance on the corpus in Serbian language [3] (section 1), then explore the content of NCD library and accordingly suggest two hierarchies of classes for hierarchical classification (section 2) and finally give a detailed plan on how the proposed method can be applied to the digitized documents library (section3).

Hierarchical classification of documents in Serbian language – an example

Method. One of the methods that can be used for hierarchical classification is Structural Support Vector Machines method (SSVM) which represents the generalization of Support Vector Machines (SVM) method for structural output. SSVM method has been introduced in 2004 [4] as one generalization of the SVM method on structural output, for example array, tree, directed acyclic graph etc. Before we describe this method, we will give a brief overview of the SVM method in the context of text classification. Creating a binary classifier for text classification traditionally implies choosing certain text properties and coding them as the input vector $\boldsymbol{x}$, then learning the discriminant function $F(\boldsymbol{x})$ and finally classifying the text represented by $\boldsymbol{x}$ according to the sign of the function $F(\boldsymbol{x})$. It is commonly assumed that the function $F(\boldsymbol{x})$ is linear so we can represent it in the form $F(\boldsymbol{x}) = \langle \boldsymbol{\omega}, \boldsymbol{x} \rangle$, where $\boldsymbol{\omega}$ is vector of learning parameters and "$\langle ... \rangle$" denotes scalar product. Since the vector space of the input vector $\boldsymbol{x}$ is usually not adequate for efficient mathematical techniques, it is often mapped into another space using function Ψ and so we write function $F(\boldsymbol{x})$ in the following way: $F(\boldsymbol{x}) = \langle \boldsymbol{\omega}, \Psi(\boldsymbol{x}) \rangle$.

In binary classification, the data are classified only in two classes and so the output Y can take values from the set $\{-1,1\}$. In that way, binary classifier answers the question "Does the document belong to a certain class or not?". In case when we have more than two classes and when they are hierarchically related, the question we need to answer is changed to "Which class does the document belong to?". This question is out of the scope of binary classifiers and we need to turn to methods that predict structural output, namely SSVM. In the basic SVM method, output $\boldsymbol{y}$ is determined according to the sign of the discriminant function, that is $sgn(F(\boldsymbol{x})) = \boldsymbol{y}$, where $\boldsymbol{y} \in \{-1,1\}$. It would be ideal if we could follow a direct analogy and find discriminant function $F(\boldsymbol{x})$ 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 a function, we turn to the next best solution and create a function $F : \boldsymbol{X} \times \boldsymbol{Y} \rightarrow \boldsymbol{R}$ that measures how well the output $\boldsymbol{y}$ corresponds to the input $\boldsymbol{x}$. In this generalization, the discriminant function becomes the function of two arguments, input and output, $F(\boldsymbol{x}, \boldsymbol{y})$, where the output can represent array, tree, graph, etc. The function F should have higher value when the output $\boldsymbol{y}$ corresponds well to the input $\boldsymbol{x}$, i.e. it is more similar to its real class, and lower otherwise. In other words, we want that value of $F(\boldsymbol{x}, \boldsymbol{y})$ for a training example $(\boldsymbol{x}, \boldsymbol{y})$ is larger than $F(\boldsymbol{x}, \boldsymbol{y}')$ for any other output $\boldsymbol{y}'$. If we denote set of all possible outputs by $\boldsymbol{Y}$, SSVM predicts the output (class) that corresponds best to the input (text), i.e. it predicts the output vector $\boldsymbol{y}$ that maximizes the value of the function F for a given input vector $\boldsymbol{x}$. To be more precise, SSVM predicts output based on the following equation: $\arg\max_{\boldsymbol{y} \in \boldsymbol{Y}} F(\boldsymbol{x}^*, \boldsymbol{y})$. Analogously with SVM, in SSVM we also assume that function F is linear in $\boldsymbol{\omega}$, as well as that pair of vectors $(\boldsymbol{x}, \boldsymbol{y})$ can be more suitably represented by mapping into some other space using a function Ψ. Therefore, we can write function F as $F(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{\omega}, \Psi(\boldsymbol{x}, \boldsymbol{y}) \rangle$.

Function Ψ represents joint input-output vector for one input-output pair $(\boldsymbol{x}, \boldsymbol{y})$ and its form is customized for each individual application. For example, SSVM can be used in predicting a derivation tree in the given formal grammar and in that case, input vector $\boldsymbol{x}$ would represent vector of words that appear in a sentence, output vector $\boldsymbol{y}$ would represent a derivation tree in the formal grammar and function $\Psi(\boldsymbol{x},\boldsymbol{y})$ would be a joint representation of the sentence and its derivation tree. One way to implement such vector would be to create a vector where each element corresponds to one of all possible rules of the grammar, and its value would be equal to the total number of occurrences of that rule. This representation is illustrated by the example on Figure 1.

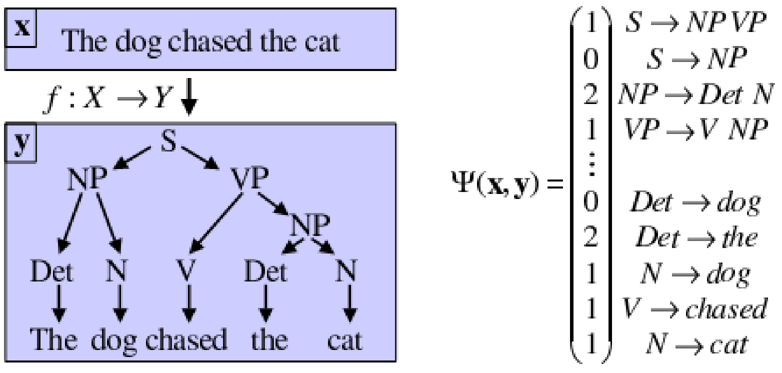

Figure 1: Example of joint representation of input vector $\boldsymbol{x}$ and output vector $\boldsymbol{y}$ [4].

In the example shown on Figure 1, vector $\boldsymbol{x}$ consists of the words from the given sentence ("The dog chased the cat"), and vector $\boldsymbol{y}$ represents a derivation tree in the given formal grammar. Vector $\Psi(\boldsymbol{x},\boldsymbol{y})$ denotes that, in the sentence derivation, rule *S→NP VP* has been applied once, rule *S→NP* zero times, rule *VP→V NP* once and so on, rule *N→cat* once. The vector $\Psi(\boldsymbol{x},\boldsymbol{y})$ contains number of occurrences for each rule in the grammar, even when it is equal to 0. The order of the rules in the vector $\Psi(\boldsymbol{x},\boldsymbol{y})$ is determined by the order that the rules appear in the grammar.

There are several variations of SSVM method [5]. In the proposed method for hierarchical text classification [3], we used the so-called 1-slack margin rescaling formulation. 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 $\boldsymbol{\omega}$ depending on a positive constant C that controls the trade-off between minimizing training error and maximizing the margin. The described algorithm has polynomial complexity by the number of training examples, which has been proven in [5].

Dataset. The proposed method for hierarchical text classification [3] has been applied to a portion of the Ebart corpus which represents the largest digital documentation of newspaper texts in contemporary Serbian language. Ebart corpus has been created in 2003 and up to nowadays it has warehoused more than 2.000.000 texts from the printed media. In the last version of Ebart archive, the documents are 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 SSVM method on the problem of hierarchical text classification, we extracted a

hierarchically organized sub-corpus of "flat" Ebart corpus by choosing all the articles from the daily newspaper "Politika", published in period 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. We named the corpus obtained EbartHier, represented on Figure 2.
The process of constructing a tree-like hierarchy tracked the following steps:

1. First level of the hierarchy is consisted of leaf nodes. Each leaf was assigned with the topic that corresponds to columns mentioned above
2. Second level of the hierarchy is consisted of the inner nodes connecting pairs of leaf nodes by the similarity of the assigned topic. For example, inner parent node *Sport* was established for leaves *Basketball* and *Football*. In this way, classes Politics and society, Economy, Culture and science and Sport have been created.
3. Third level of the hierarchy consists only of the root node.

Each node in the hierarchy represents one topic and consequently one class that can be assigned to a document. 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. It is notable that the distribution of documents by classes is extremely uneven (see Figure 2). All the documents in the corpus are represented in Latin alphabet using UTF-8 code scheme.

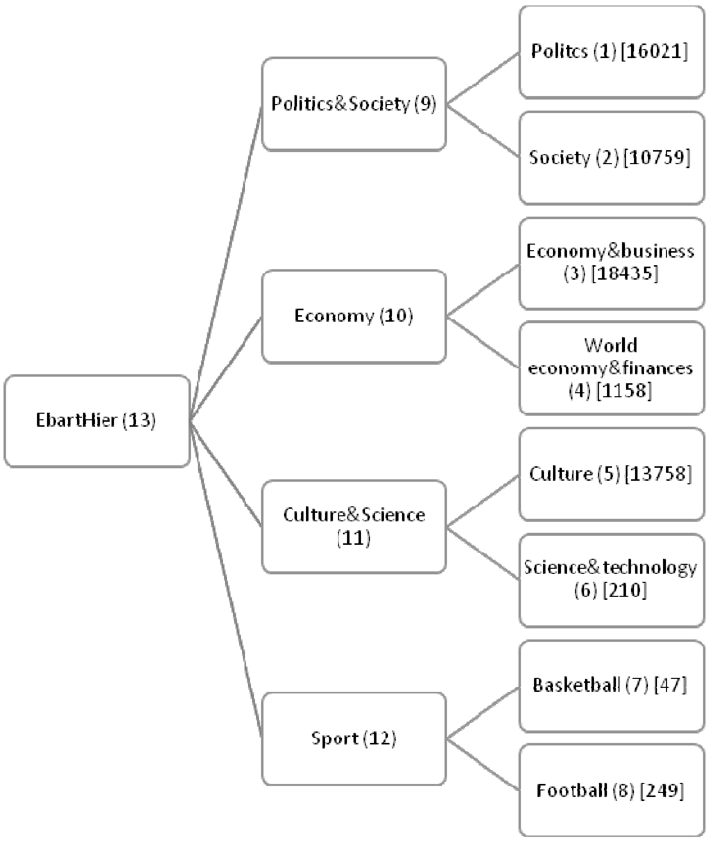

Figure 2: Hierarchy of classes in corpus EbartHier. For each node in the hierarchy, its enumeration is displayed in round brackets and, in addition, next to leaf nodes the number of texts assigned the corresponding class is displayed in square brackets.

Data representation. In order to apply SSVM method to the hierarchical classification of the corpus described above, it was necessary to determine the representation of the input $\boldsymbol{x}$ (text) and the output $\boldsymbol{y}$ (class), as well as their joint representation, function $\Psi(\boldsymbol{x},\boldsymbol{y})$.

In context of text classification, different text features can be used as input attributes for the classifier. The most widely used features include different types of n-grams. An n-gram of the string $s=s_1s_2...s_N$ (for natural numbers n and N) is defined as any substring of adjacent symbols of the string s of length n. 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. Although byte n-grams sometimes do not have specific meaning, especially for human kind (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 the proposed method, we used byte n-grams.

Each document is represented by its n-gram vector, i.e. a histogram of all byte n-grams that appeared in the dataset. n-gram vector for each text is coded by the input vector $\boldsymbol{x}$ whose dimension corresponds to the total number of byte n-grams that appear in the dataset. The value on each position in input 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 [6]:

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

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

The output vector for each class is represented as vector of nodes in the corresponding hierarchy of classes. Each position in vector $\boldsymbol{y}$ 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. For example, in EbartHier hierarchy (Figure 4), class *Culture* would be represented as (0,0,0,0,1,0,0,0,0,0,1,0,1) [3].

Dimension of the vector $\Psi(\boldsymbol{x},\boldsymbol{y})$ is $p \cdot r$, where p is the number of different n-grams, i.e. dimension of the vector $\boldsymbol{x}$, and r is the number of nodes in the hierarchy. In that way, each node gets its block of size p in the 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 $\boldsymbol{x}$, otherwise. For example, if document $\boldsymbol{x}$ belongs to class *Culture* (EbartHier), represented as (0,0,0,0,1,0,0,0,0,0,1,0,1) then the joint representation will be the following:

$$\Psi(\boldsymbol{x},\boldsymbol{y}) = [\boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{x},\boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{x},\boldsymbol{0},\boldsymbol{x}]$$

where **0** is the zero vector $\underbrace{[\boldsymbol{0},\ldots,\boldsymbol{0}]}_{p\ times}$, and vector $\boldsymbol{x}$ corresponds to the input vector.

Model development. The proposed hierarchical classifier was constructed by adjusting publicly available and free SVM^{struct} framework for SSVM method [7]. SVM^{struct} is available in several programming languages and for this purpose we used its implementation in programming language C. Depending on the length of byte n-gram (n from 2 to 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*). For each representation, we divided documents on training set and test set in proportion 2:1 and performed 10-cross validation on each training set in order to find the optimal value of parameter C. In each representation, training sets and test sets consisted of the same documents so that the performance of classifiers could be compared on them.

Evaluation and results. The performance of the proposed method was measured using micro-F1 measure, which represents one of the common ways of generalizing F-measure, defined for binary classification, to the problem of multiclass classification, including hierarchical classification as its special case. F-measure is defined as the harmonic mean of precision and recall:

$$Precision = \frac{TP}{TP+FP}, Recall = \frac{TP}{TP+FN}, F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

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. One way to average these values is to find so-called micro-average by calculating values for *TP*, *TN*, *FP* and *FN* for each class individually and then to calculate values ***TP***, ***TN***, ***FP*** and ***FN*** by summing all *TP*, *TN*, *FP* and *FN,* respectively, for all the classes. At the end, we calculate the measures for the sums ***TP***, ***TN***, ***FP*** and ***FN.***

We applied SSVM method on 24 different representations of EbartHier corpora, one for each byte n-grams length from 2 to 7 and for each of 4 different tf-idf measures. 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} to 10^{2}, with step 10. The results are shown in Table 1.

EbartHier (60637)				
N	**Classic**	**Log**	**boolean1**	**boolean2**
2	44.21%	81.75%	82.04%	82.45%
3	45.42%	88.34%	88.86%	89.24%
4	48.61%	89.39%	89.58%	**90.17%**
5	49.84%	89.46%	89.57%	89.70%
6	50.65%	89.84%	89.47%	89.78%
7	51.01%	89.88%	89.68%	89.69%

Table 1: Performances of hierarchical classifier for different representations of EbartHier, expressed by micro-F1 measure. Total number of documents is shown in round brackets. Best result is emphasized in boldface.

SSVM hierarchical classifier shows worst performance for classic tf-idf measure. Differences in performance for a specific length of byte n-grams among different measures (excluding classic), range between 1% and 2% for the Ebart corpus. On the other hand, differences in performance for a specific tf-idf measure (excluding classic) among different lengths of byte n-grams, range between 3% and 8%. The results obtained suggest that different combinations of type of tf-idf measures and byte n-gram lengths have influence over the total results. Best result obtained (90.17%) outperforms best known result on Ebart dataset (88.5%, [8]).

Framework for hierarchical classification of NCD digital library

NCD digital library consists of around 3600 different documents organized in 11 communities. Additionally, documents in some communities are further divided in collections and subcommunities. This classification is flat, i.e. communities and collections are not related in any way. The communities in the NCD digital library include "Arts and humanities", "CD library", "Mathematical sciences" and so on. Community "Mathematical sciences" is the most numerous containing around 2700 documents in six subcommunities: Books, Collected works, Doctoral dissertations, Master of science works, Master works and Scientific works. Documents from several subcommunities (all except Books and Collected works) are divided further by topic on Astronomy and Geoscience, Mathematics and Computer science. At present, topics are broadly defined and finer search by topics is lacking (the link for "Search by topic" exists on elibrary.matf.bg.ac.rs but to our knowledge it is not functional). One way to implement this functionality is by developing a model for hierarchical classification. Class hierarchy can be based on different document features – type of document, area, subject, author, time period etc. An example of such a hierarchy is represented on Figure 3. It contains three inner classes, defined by existing subcommunities, and several leaves assigned more specific areas of each science field.

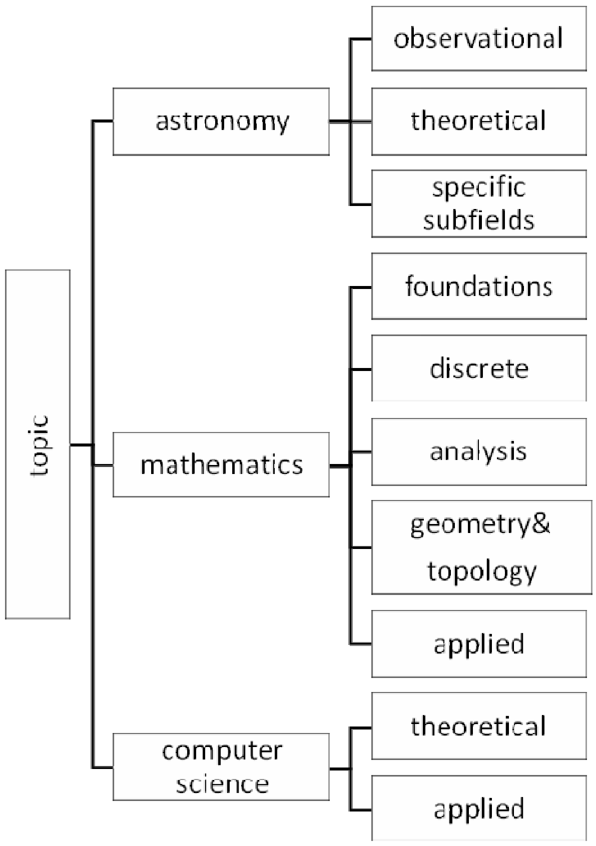

Figure 3: An example of hierarchy of classes by topic.

Another set of classes with hierarchical relationship refers to the issue date, more precisely the issue year. NCD library contains texts from mid-18th century to the current year with the majority of documents issued in 19th and 20th century. This fact inspired the creation of one possible hierarchy displayed on Figure 4.

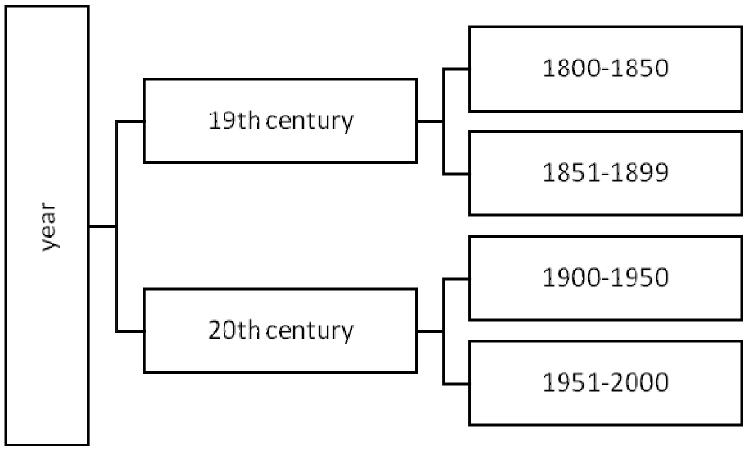

Figure 4: Example of hierarchy of classes by issue year.

Classification model for such hierarchy could be constructed over all documents of the library (issued in 19th and 20th century) or on a particular subset. It would be especially interesting to compare the performances of this classification model trained on entire library corpus, regardless the topic, with the model trained on texts with topic Mathematics only, in order to check whether the presence of documents regarding the development of computer science in the 20th century would have any influence on the quality of the hierarchical classification.

Application of SSVM method for hierarchical classification on NCD library documents

SSVM method for hierarchical classification has the potential to be applied to the documents of the NCD digital library. The library would highly benefit from such classification since it would enable new forms of useful search. In order to make it possible, the following protocol should be followed:

1. SSVM method is language independent, but all documents in the corpus used for training and testing need to be in the same language. That is why the first step would have to be extraction of documents in one language, perhaps in Serbian since these documents are the most numerous
2. Prior to any automatic classification of documents, it would be necessary to convert each document from .pdf format into machine-readable .txt format. OCR can be a technology of choice, especially since byte n-gram representation is insensitivity to errors that could occur in the process of automatic character recognition.
3. A hierarchy of classes needs to be chosen (e.g., previously mentioned topic hierarchy, or time hierarchy, or any other type). In case of topic hierarchy (Figure 1), an additional annotation is needed, i.e. each document should be manually assigned a more specific topic (leaf level in hierarchy). Depending on available documents and their classes, the proposed hierarchies can be adjusted in the sense that certain nodes can be added or removed. Potential problem that might occur is unbalanced number of texts per class. SSVM hierarchical classifier has already been confronted with this challenge and it has shown to be robust in case when number of documents per class is unbalanced (Figure3, class *basketball*).
4. It is necessary to extract byte n-grams for each text. A useful tool for this action can be Text::Ngrams [9] which generates the array of byte n-grams that the document contains as well as how many times each byte n-gram appears.
5. Based on the byte n-grams produced in the previous step, we need to preprocess data in the form required by the SSVM classifier
6. After all activities regarding document preparation, we can proceed to training SSVM method for the prepared documents on the chosen hierarchy of classes and to testing its performances afterwards.

Conclusion

In this paper, we have presented results of the application of SSVM method to hierarchical classification of documents in Serbian language. Since this method has shown to perform well on Serbian corpus, we expect that it would achieve similar results when applied to documents of the NCD digital library. We offered a proposal of two hierarchies of classes as well as a protocol for the NCD library preprocessing necessary in order to start applying the proposed method to digitized documents. The project would improve usefulness of the library in a way that it could contribute to finer search of its documents by more specific classes defined in leaves of the hierarchies.

Acknowledgement

This work was supported by the Serbian Ministry of Education and Science (project 174021).

References

1. Manning, C., and Schütze, H., Foundations *of Statistical Natural Language Processing*, MIT Press. Cambridge, MA. 1999.
2. Sun, A. and Lim, E., *Hierarchical text classification and evaluation*, Data Mining, Proceedings IEEE International Conference, 2001, 521–528.
3. Kovacevic J. and Graovac J. , *Application of a Structural Support Vector Machine Method in N-gram based text classification in Serbian*, Infotheca Journal for Digital Humanities. To appear.
4. Tsochantaridis I. et al., *Support Vector Machine Learning for Interdependant and Structured Output Spaces*", Proc. of the Twenty-first International Conference, 2004.
5. Joachims T. et al., *Cutting Plane Training of Structural SVMs*, Machine Learning Journal, 77:1 (2009), 27–59.
6. Manning, C., Raghavan, P., and Schutze, H., *Chapter 6: Scoring, term weighting, and the vector space mode,* Introduction to Information Retrieval, Cambridge: Cambridge University Press (2008)
7. https://www.cs.cornell.edu/people/tj/svm_light/svm_struct.html
8. Graovac, J., *Serbian text categorization using byte level n-grams*, Proceedings of CLoBL 2012: Workshop on Computational Linguistics and Natural Language, 5th Balkan Conference in Informatics, 2012, 93–97.
9. Kešelj, V., Peng, F., Cercone, N. and Thomas, C. "*N-gram-based author profiles for authorship attribution*", Proceedings of the conference pacific association for computational linguistics, PACLING, 3 (2003), 255–264.