N-GRAM-BASED SERBIAN TEXT CLASSIFICATION

Petar Prvulović[1], Nemanja Radosavljević[1], Dušan Vujošević[1], Dhinaharan Nagamalai[2], Jelena Vasiljević[1]

[1]School of Computing. Union University, Belgrade, Serbia
[2]Wireilla, Australia

ABSTRACT

Natural language processing is an active area of research which finds many applications in variety of fields. Low-resource languages are a challenge as they lack curated datasets, stemmers and other elements used in text processing. Statistical approach is an alternative which can be used to bypass lack of rule-based implementations. The paper presents a model for classification of unstructured text in Serbian language. The model uses n-gram-based stemming to create document attributes vectors. Vectors are created on 3-, 4- and 5-grams. Vector reduction is tested on two criteria: n-gram entropy and number of occurrences, and two lengths: 1000 and 2000 n-grams. The support vector machine is used to classify documents. The model is trained and tested on a dataset collected from a Serbian news portal. Classification accuracy of over 80% is achieved. The presented model provides a good basis for range of applications in business decision automation for low-resource languages.

KEYWORDS

N-gram stemming, Serbian language, Unstructured natural text categorization

1. INTRODUCTION

Low resource languages lack the proper dictionaries of prefixes, suffixes, stop words and other elements needed for stemming and lemmatization, as well as other resources to properly train existing language processing algorithms. Some of the existing approaches circumvent the lack of stemmers by translating documents to English using Google translate and similar tools. These approaches provide usable results but depend on an external system, and information on specific writing style of a document might be lost in translation. Other approaches are based on statistical text analysis, which also provide useful results, however these approaches are claimed to be slower and resource consuming [1].

The paper presents a model for classification of unstructured text in Serbian language. The proposed solution relies on statistical n-gram text analysis which extracts category-specific n-grams to train a classification model. The model is trained on a document set scraped from a Serbian news portal. Hyperparameters of the proposed model include the following: n-gram length of 3,4 and 5 letters; document vector reduction - the criteria to select significant n-grams using n-gram entropy and number of occurrences; and vector length of 1000 and 2000 n-grams.

There are several advantages of the proposed model. The proposed model can be used for any natural language. The model is fully self-contained and independent as it is based on a training dataset and doesn't rely on external systems such as Google Translate. The text is processed as-is

David C. Wyld et al. (Eds): MLNLP, NWCOM, DTMN, ASOFT, SIGPRO, AIFZ, CSITY, CLSB - 2023
pp. 189-201, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.131613

- no translation or any other altering is needed during the process, which allows preserving writing styles specific to categories. The model does not require developing a language model and rules for prefix, suffix and stop words removal and text normalization into canonical form.
The paper is structured into Sections as follows: Section 2 outlines the problem and highlights some existing work in this area, and outlines advantages and drawbacks of existing and proposed approaches. Section 3 describes the dataset used to train and test the proposed model, and how the dataset was formed. Section 4 describes the concept of the proposed model and model components. It also provides a brief intro in KNIME, the platform used to create the model. Section 5 explains the process of feature extraction based on n-grams and how n-grams compare to natural language stemming. Section 6 describes the vector reduction process and two criteria used for n-gram selection: n-gram entropy and number of occurrences. Section 7 explains how the model accuracy was tested and provides testing results and possible improvements.

2. RELATED WORK

The existing work in Serbian language corpus can be found in the Literature [2,3,4,5,6,7,8]. Annotation process of contemporary Serbian is described in [2]. The corpus words contain lemma, derivational stem and prefixes, infixes and suffixes. Corpus is compiled into SrpKor2013 [3]. SrpLemKor is a subset of SrpKor2013 which contains 3.7 million of lemmatized words [3]. Serbian WordNet is a database of words grouped in sinsets – sets of synonyms. In [4] text classifiers are developed and the author states that the main drawback of Serbian WordNet is that the number of words and sinsets is not comparable to English WordNet.

Text pre-processing is an important step in unstructured text classification. It aims to remove stop-words, special characters and other elements that can be considered as noise from text and normalize the words into canonical form by extracting the root words. The pre-processing uses one preferred root word instead of all the possible synonyms. It is performed in several stages: stop-word removal, punctuation and digits removal, stemming and lemmatization. Stemming is a process used for normalizing different variations of words into the same form, called stem, by removing affixes. Lemmatization is a process used for normalizing synonims into the same word. The mentioned resources can be used for such purpose, though processes which implement this approach suffer from need of extensive word databases and their weak point is the word database incompleteness.

Stemming doesn't require dictionaries as it doesn't deal with synonyms. The process of removing prefixes and suffixes can be to some extent described by a set of rules. Available linguistic resources can be used both to generate rules and to test stemmer accuracy. In [5] author evaluates the efficiency of available stemming algorithms for Serbian. Three of them are based on suffix subsumption [6,7] while one relies on regular expressions [8].

Statistical text analysis [1] based on n-grams enables further text analysis and extraction of category-specific parts of words regardless of their lexical form and is linguistically independent which makes it suitable for the Serbian. Serbian language is morphologically rich as one word can be written in several forms using a variety of affixes for conjugation, declination etc. Repetitive use of a word is expected to statistically emerge n-grams that contain the word's stem, as opposing to n-grams that contain prefixes and suffixes, effectively extracting stems, or parts of stems, thus allowing the implementation of stem-based document feature vectors without lexical resources, independent of language.

3. DATASET

Dataset is formed by scraping the news from Serbian news portal. The collected documents are classified in 6 categories: Politics, Region, Society, Chronic, Organized crime, EU.

The scraper is implemented as a PHP script, which fetches, parses and traverses through lists of paginated content. The starting pages for each category are set manually. Upon analyzing the DOM tree of a category page, hyperlinks to news and next page were identified. Analysis of the DOM tree of a single news page enabled finding the path to title, short intro and full document text. The scraper PHP script uses PHPQuery library to parse the HTML page into a DOM tree and query the DOM for needed hyperlinks and document segments. The scraper script was initiated with a starting URL – a single category page, and left to iterate through category pages, collect all the new links, and then fetch and parse each news item and save it as a document. Category pages were handpicked manually.

Resulting dataset contains32107 documents. For each document following attributes were collected: title, short intro, full text, category. HTML tags and consecutive whitespace characters were stripped from documents in order to keep the plaintext content only. Dataset is stored as a CSV file.

Two of the common challenges in supervised learning are the amount of training data and class imbalance [9]. Table 1. lists the number of documents for each class, their respective proportion of the dataset, and total number of documents in the dataset. There is no formal definition of when a dataset is imbalanced. An accepted rule, noted in [10], says that a dataset with two classes is mildly imbalanced if the proportion of a minority class is 20-40% of the dataset and extremely imbalanced if the minority class is <1% of the dataset. Two classes are less represented, as can be seen in the table. The least represented class contains 2753 documents, which is 55% of 5000 – highest and most common number of documents in classes of the dataset, so the dataset can be considered as balanced.

Table 1. Heading and text fonts.

Class	**Number of documents**	**% of total**
Politics	5000	19%
Region	5000	19%
Society	5000	19%
Chronic	5000	19%
Organized crime	2753	10%
EU	3554	14%
Total	26307	100%

4. MODEL DEVELOPMENT

Classification can be approached in two ways: supervised and unsupervised classification. Supervised classification requires that the training dataset is classified upfront. Unsupervised classification groups the samples which are similar in some criteria and extracts the set of classes. The developed model uses supervised classfication. Dataset classes are news categories.

The presented model utilizes Support Vector Machine (SVM). Other algorithms, such as k-nearest neighbors, can be used for the same purpose. The model is designed in such a way that the SVM node is interchangeable with other classification algorithms.

For all classification algorithms it is necessary to associate a feature vector to each sample. In this way, each sample is mapped in a *n*-dimensional vector space. Thus the classification algorithm can, in its specific way, recognize in which zone the sample is located, and associate the estimated best fit class. The assigned *n*-dimensional feature vector is determined by the choice of characteristics considered relevant to the classification. Characteristics can be binary – if some specific word exists in a document, numeric – how many times a specific word appears in a document, or some other measure can be used. Algorithms determine the boundaries of vector classes based on the document feature vectors whose classes are known, and form classification rules based on that.

Figure 1. shows a schematic representation of the classifier model. The training phase extracts the document features, creates document feature vectors and forms the classification rules based on the given data sample – training set. The prediction phase gets a document, extracts the features, creates a document feature vector, and finds the class which fits best to the document.

Standard way to measure a classifier's performance is to split a part of the dataset into a testing set, and compare the classifier output to actual classes. The classifier performance is measured and results are shown in Section 7.

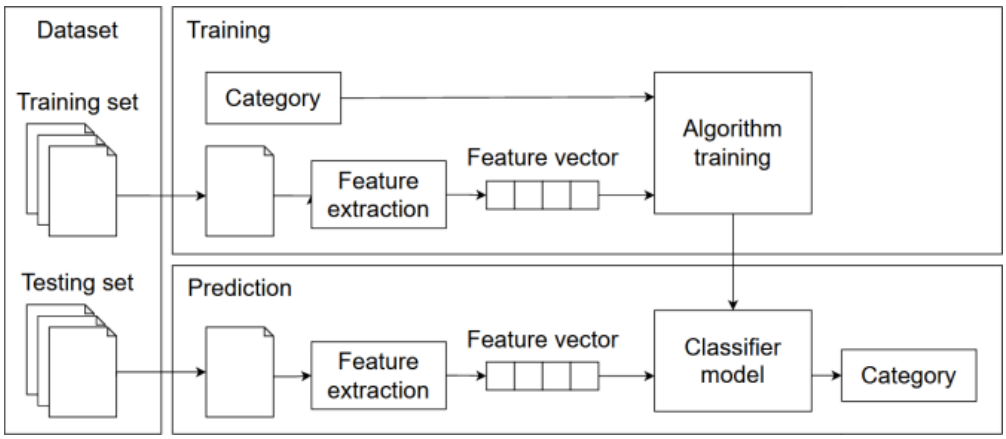

Figure 1. Schematic representation of the classifier model

4.1. Model Implementation in KNIME

The model is developed in KNIME[11]. KNIME enables creation of a model as a graphical scheme, using programmable parts of code represented as graphical blocks-nodes, whose inputs and outputs are connected in a flow, in a drag-and-drop fashion. Supporting library is rich in ready-to-use nodes, having both simple and complex nodes, such as data source connection, data type converters, statistical formulas, data mining methods, machine learning algorithms etc. KNIME model typically starts with nodes for data acquisition, such as CSV Reader or MySQL connector; and ends with nodes that store data or display data in tabular form or in charts, like CSV Writer, Histogram or Table view. The flow is not necessarily linear, and allows for multiple output nodes. Custom nodes can be created in Java.

Creating a model in the form of a flow, by dragging and connecting graphical nodes, removes the need for the model author to be a skilled programmer, and allows the author to focus on the model and data processing methods. This approach makes it easier for specialists in other areas to create analytical models and contribute to a team without having additional team members to support them by writing and/or correcting code for their models.

5. FEATURE EXTRACTION

Supervised classification requires assigning a feature vector to each document. Feature vector is treated as a point in *n*-dimensional space, *n* being the vector size. During the training phase, classified documents are mapped into *n*-dimensional space at the positions defined by the feature vector thus creating a set of points where each point represents one document of a specified class. Support Vector Machine (SVM) algorithm takes the set and finds the boundaries for each class, effectively producing classification rules. Classification of new documents is a process of extracting the features of that document, creating a feature vector, mapping the document to a point in n-dimensional space and deciding which area that point falls into, thus assigning a class to that document.

In unstructured documents classes can be considered as specific topics and it is reasonable to expect specific words to appear more frequently in some of the classes. Appearance of a word specific to some class in a document makes that document a bit more biased towards that class. A word in natural language can be written in multiple forms, depending on the morphology of the language. Word stem, or root, is a part of a word that carries lexical meaning. Word is created by adding prefixes and suffixes to stem. In order to properly extract document features words should be reduced to stems, so as to avoid differentiation of words by their form instead of meaning. Proper word stemming requires a set of rules for the target language. In case of the Serbian language such rules exist but their quality is to be tested [4,5,6,7] and they are yet to be implemented in data processing tools such as KNIME, which is a major flaw when processing documents in less supported languages.

5.1. N-gram-based Stemming

N-gram based statistical stemming works on a presumption that the n-gram which contains a word stem will appear more often than n-grams which contain an affix and part of the stem. One word can appear in multiple forms (singular and plural etc). Multiple forms of one word differ in affixes but have the same stem. Appearance of several forms of one word expectedly yields the n-grams which contain the stem of the word and a variety of n-grams which contain affixes. It is expected to encounter significantly more appearances of n-grams which contain the stem. This provides the basis for stem extraction. Classification model can then be trained using stem appearance specific for classes. This works best when a stem is the same length as sampled n-grams. If the stem is shorter, parts of affixes will be included, which degrades the precision to some extent. If the stem is longer, a part of the stem will be sampled, but that part should statistically stand out in similar fashion as the stem would.

The proposed model was tested for 3, 4 and 5-grams. N-gram length is varied in order to test which length works best. Most of the stop words in Serbian are one or two letters, which get ignored so the n-gram stemmer works as a stop word filter to some extent.

Table 2 illustrates the stem extraction based on n-grams of the length 3,4 and 5 letters on four example words. Example words are: *učitaj* – load (data), which is in Serbian formed from verb read – "to read in"; *čitamo* – (we) read; *čitanka* – a reading book; *mučitelj* – torturer. First three words have the same stem – *čit*, while the last word's stem is *muč*. As can be seen in Table 2, 3-gram *čit* is present in all cases, which would be a wrong assumption for the last word. 4-grams seem more promising as *čita* is present in the first three words, but not in the last word, where *muči* is extracted. 5-grams provide bad results in this case, as the first three words do not have any common n-grams. This example does not imply a rule, and n-gram lengths need to be

decided experimentally, though in the results section it can be seen that 5-grams produced the worst results.

Example needs to be seen in the context of classes also. If the first word appears in documents of the class A, the second and third in class B, and the last word in class C, it can be concluded as a rule that 3-gram *uči* is specific to class A, *čit* to class B, and *muč* to class C. We can conclude similar for 4-grams: *učit*, *čita* and *muči* could be considered specific for classes A,B and C, respectively.

Table 2. Example of word n-grams

Word	**3-grams**	**4-grams**	**5-grams**	**Class**
učitaj	uči,čit,ita,taj	učit,čita,itaj	učita.čitaj	A
čitamo	čit,ita,tam,amo	čita,itam,tamo	čitam,itamo	B
čitanka	čit,ita,tan,ank,nka	čita,itan,tank,anka	čitan,itank,tanka	B
mučitelj	muč,uči,čit,ite,tel,elj	muči,učit,čite,itel,telj	mučit,učite,čitel,itelj	C

5.2. Feature Extraction

Feature extraction for a single document results in a vector containing every distinct n-gram which appears in the document, and the number of occurrences of that n-gram in the document. Feature extraction requires some basic text-preprocessing in order to normalize the text. The entire process is as follows:

- data is read from CSV file
- case is converted to lowercase
- strings are converted to documents
- classes are assigned per categories in the last column
- numbers are erased
- punctuation is erased
- words shorter than n-gram are removed from documents
- n-grams are extracted and counted in each document, thus assigning the document a full-length feature vector.

The order of the steps is dependent on KNIME node implementation, as nodes expect some kind of data on input and provide a specific output. Case converter node works on string columns, while Number filter works on document columns, hence the order of nodes in the flow.Figure 2. shows the KNIME diagram of the entire process up to the point of n-gram extraction.

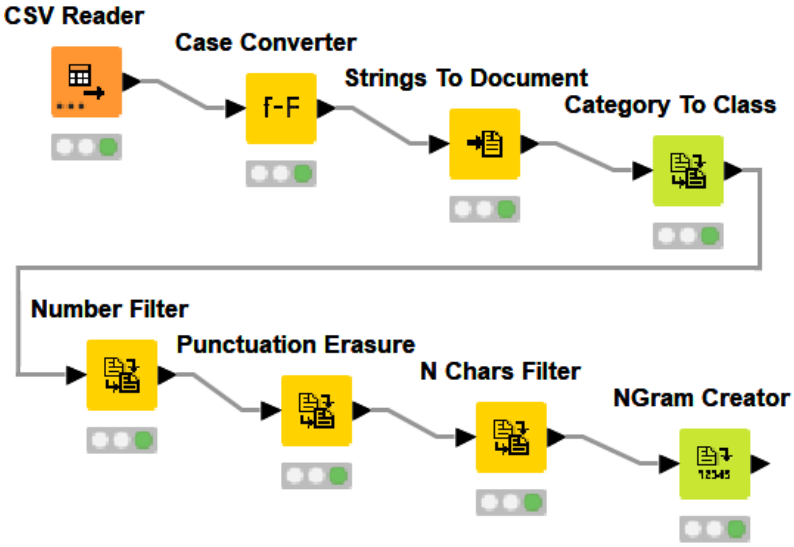

Figure 2. KNIME workflow for extracting document features and assigning a full-length feature vector

6. VECTOR REDUCTION

As each document is assigned with a feature vector and positioned into the *n*-dimensional space by values in that vector, vectors need to be of the same size and have the same dimensions. The vectors can be too long for the classifier as they include all of the n-grams gathered during the training phase. The vector reduction is performed as a step which reduces the number of dimensions of the vector space. The vector reduction is based on selecting the most relevant n-grams to get the best classification results. N-gram relevancy is measured in two ways: n-gram entropy and number of appearances. Based on these measures, best rated n-grams were included into the reduced feature vector. Vector size was tested for 1000 and 2000 n-grams. Document vectors created in this way are passed to the SVM trainer as the input for training, and are used for document classification based on the generated SVM classification rules.

Classification based on n-grams specific to classes is proved useful in a variety of situations[12,13,14].A major flaw is that such classification could include the affixes which statistically stood out, instead of just word stems. On the other hand, natural text tends to be written in a consistent form, regarding the passive/active style, tense, etc. which might be specific to a class, in which case such affixes bear the useful information.

To perform the n-gram scoring, pre-steps were executed to provide the same input for each of the three scoring methods used. Pre-steps include text case conversion, number filtering, punctuation erasure and short words filtering. After the Ngram creator node generates a list of n-grams in each document in the training set, each n-gram occurrence is assigned with a class of the document it was extracted from. Using the GroupBy node a number of documents for each class is computed in one branch of the workflow. In another branch, Math Formula node is used to assign the Boolean indicator of a document containing the n-gram, and fed into the Crosstab node to cross the data by n-grams and columns and provide the number of documents containing an n-gram in each class. Excessive columns are removed using the Column Filter node, which leaves the n-gram, class, and relative n-gram incidence in the class expressed in range [0,1] which can be thought of as the probability of n-gram appearance in a class. The last column is referenced as NP in later formulas. That table is joined to a table containing a number of documents in each class. The resulting table is used as input in scoring methods.

6.1. Number of Occurrences

Scoring the n-gram relevancy by number of occurrences is simply performed by counting the occurrences of an n-gram in each of the classes, and taking the maximum value. N-grams are then sorted by the maximum number of occurrences in a class in descending order, and first *n* n-grams are included in the feature vector, *n* being the length of the feature vector. The implementation in KNIME is easy and uses GroupBy, Sorter and RowFilter nodes on top of the previously computed n-gram appearance in documents assigned to classes

6.2. N-gram Entropy

Scoring the n-gram relevancy by entropy is based on the idea of entropy as a measure of uncertainty of an event's outcome. Entropy is highest if an event has equally likely outcomes, as there is no outcome that is more specific for the event and can be expected more likely than any other. If one outcome is more specific to the event, the entropy of that outcome is lower. Predicting the outcome based on entropy is more accurate for low entropy outcomes. Entropy is 0 when there is one outcome with probability of 1, i.e. the event has an expected, predefined outcome. In the context of n-gram scoring, n-gram is the event and document class is the outcome. Ideal n-gram is the one which appears only in one class and has 0 entropy, as such n-gram is a strong signal that the document containing it is in that class.

The model is simplified in such a way that it uses the n-gram presence in a document as a Boolean indicator, which ignores the incidence of n-gram appearance in documents. Each n-gram has a total number of documents it appears in each of the classes. Number of documents in each class is known. Dividing these two values gives the relative n-gram frequency in a class, in range [0,1]. By scaling the sum of relative frequencies to 1 and subsequently scaling the relative frequencies a discrete random variable S was defined. The outcomes of S are document belonging to classes, and have a joined probabilities defined by n-gram incidence in documents in those classes.

Information entropy can be calculated in several ways. The model uses the following formula, noted in [15]:

$$H(S) = -\sum_{i=1}^{n} P(S_i)\ln P(S_i) \quad (1)$$

S is the probability system of the random variable, the n-gram incidence in classes. *n* is the total number of classes. si are events of a document belonging to class *i*. $P(S_i)$ is event probability - the probability that the n-gram belongs to class *i*.

The KNIME workflow continues on previously computed n-gram appearance in documents assigned to classes, which contains the NP column with incidence in range [0,1] - the $P(S_i)$ values. Math Formula node calculates partial sums for entropy with the following expression, according to (1):

Output = -NP/100 *ln(NP/100) (2)

which are then summed using Group By node so to get the *H(S)*, the n-gram entropy. Sorter and Row Filter nodes are then used to sort and select top 1000 and 2000 n-grams with lowest information entropy value.

6.3. Vector Size

Two vector lengths were tested: 1000 and 2000 n-grams. These lengths were chosen arbitrarily. The longer the vector the more precise classification is expected, on the cost of execution performance, which proved true in case of N-gram number of occurrences. N-gram entropy initially yielded significantly poorer results, though longer vectors provided better results, as expected. Measuring the classifier accuracy is explained in Section 7. Figure 3. shows initial classification accuracy for all three vector reduction approaches used, for both vector lengths, for 3-grams. Accuracy is expressed in range [0,1], 1 being 100% correct class recommendations. It is visible that accuracy rises with longer vectors, and that N-gram entropy yielded results below expectation.

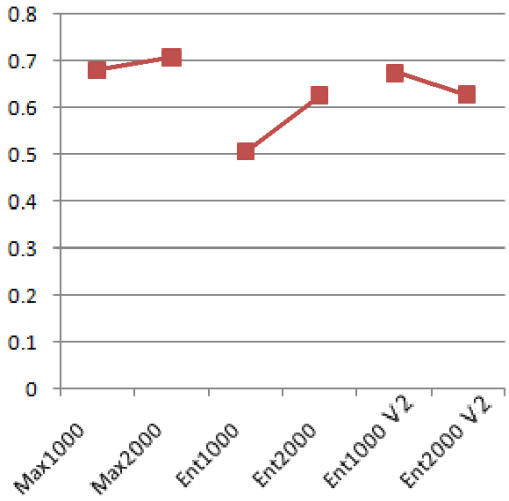

Figure 3. Classification accuracy of feature vector reduction approaches using vectors with 1000 and 2000 n-grams

N-grams that appear only once or several times have the highestclass entropy, as when n-gram occurs only once, its class incidence is 100% biased to that class, giving the zero entropy. Such n-grams are of little to no use as they appear rarely in documents and consume feature vector space, and in general case should be treated as noise. Increasing the vector length included n-grams with entropy close to 0, which appear more frequently in documents hence are more relevant, which explains the improvement in accuracy.

N-gram entropy reduction was modified to address this problem by removing the rarely occurring n-grams. For each n-gram, the highest number of appearances in a class was taken, averaged, and all the n-grams with below-average maximum number of appearances in a class were removed. Feature vector was then created by selecting the best scored 1000 and 2000 remaining n-grams. This enabled including the frequent n-grams into the feature vector. Figure 4 shows results for 3-grams after this modification. Improvement is visible, as n-gram entropy now yields best results, as initially expected. Increasing the vector length provided worse results due to inclusion of n-grams that more uniformly present in classes. Further analysis and fine tuning is possible and should be addressed in future work.

7. MEASURING THE MODEL ACCURACY

Dataset is initially split into a training set and a testing set. Documents in the testing set are used to measure the performance of the classifier by passing them to the trained classifier and comparing the classifier output with the actual class of the document. Scorer node calculates the accuracy statistics for each class by counting the correct and incorrect predictions, both in terms of document belonging and not belonging to that class. True positives, false positives, true negatives and false negatives are counted for each class, and used to calculate the overall classifier accuracy as overall accuracy and Cohen's Kappa[16].

Overall accuracy expresses the ratio of correct predictions to the total number of predictions. It is calculated by the following formula:

$$Overall\ Accuracy = \frac{TP}{TP+FP} \tag{3}$$

where TP are true positives – number of documents in a class correctly classified to that class, and FP are false positives – number of documents from other classes wrongly classified to a class.

Cohen's kappa expresses the agreement between two classifiers, in this case the expected and predicted results. It is calculated by the following formula:

$$k = \frac{p_0 - p_e}{1 - p_e},\ p_e = \frac{1}{N^2}\sum_k n_{k1} n_{k2} \tag{4}$$

where $p_0$is the ratio of number of predictions with same outcome in both scorers to total number of predictions; p_e is probability of both scorers predicting the same outcome by random chance; N is number of samples; k is number of classes; and n_{ki} is number of samples which classifier i predicted a class k. Numerator expresses the observed difference and denominator expresses the maximum difference of trained classifier and random classifier. For a random model, the overall accuracy corresponds to random chance, the numerator is 0, and Cohen's kappa is 0. For a good model, the observed difference is close to the maximum difference and Cohen's kappa is close to 1.

Figure 5. shows the charts with overall accuracy and Kohen's kappa measured for 3-, 4- and 5-grams, on feature vectors with 1000 and 2000 n-grams, for both vector reduction strategies: number of occurrences and modified n-gram entropy.

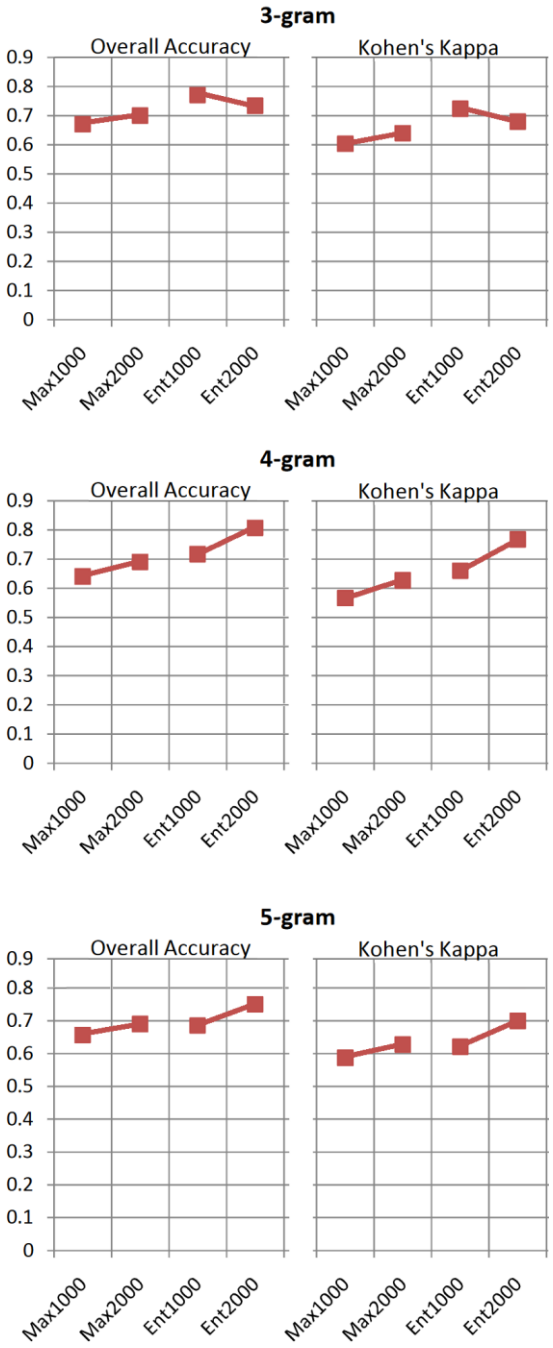

Figure 4. Model accuracy for different n-gram lengths, feature vector lengths and feature vector reduction methods

8. CONCLUSIONS

Low resource languages lack proper resources to train language processing algorithms. Stemmers for the Serbian language do exist but their quality is to be tested and improved, and implementation and support in data processing tools is virtually non-existent. It is not uncommon to bypass these limitations by using automatic translation to English as a pre-processing step. Such approaches suffer from relying on external translation services and possible loss of specific

writing styles due to loss in translation. The proposed model applies a statistical approach to text classification. Model is trained on a prepared dataset in supervised fashion, using n-grams as document features. This approach allows for a self-contained system which is not dependent on external services, dictionaries and predefined sets of language rules. As it builds on a dataset, the proposed model can be adapted to any natural language.

Classifier is developed with unstructured natural text documents in focus. The dataset was collected from the Serbian news portal, from categories which are in close topics. The classifier trained on that dataset provided useful accuracy, as can be seen in Section 7.

There could be several possible improvements of the proposed method. N-gram selection in feature vectors could be better balanced class-wise to include an equal number of vectors from each class, but as anticipated in Section 6, there are possible drawbacks and this needs to be tested and compared to the current model. Longer feature vectors can be tried to find the optimal feature vector length, which should be an automated process during the training phase. Feature vectors could also include mixed-length n-grams.

Document feature vectors are based on n-gram incidence and classification relies on n-gram appearance in documents, which provides a range of possible applications. Classifier allows for automatic categories and tags recommendation in content publishing systems, which was tested with satisfying results. N-gram incidence allows for classification based on specific writing style, as people tend to have specific vocabulary, use some words and express in some forms more than others. Future work will include testing if the proposed model can be applied to classify documents to authors which can be used to detect attacks such as email spoofing, and impersonation in general. The proposed model could prove to be helpful in categorizing semi-structured and structured documents as well, as such documents tend to have specific prefixes, suffixes, abbreviations and similar features which are distinct to document categories or tags assigned to documents.As Serbian language is a small language and not well supported both in linguistic and software resources, and the proposed classification method being adaptive to language and content, and self-sufficient, improving the method might prove beneficial in further adoption of decision support systems in Serbia.

REFERENCES

[1] Anjali,J. Ms., “A Comparative Study of Stemming Algorithms”,Technopark Publications, IJCTA-International journal of computer technology and applications, Vol. 2, No. 6, pp. 1930-1938. 2011.

[2] Utvić, M. “Annotating the corpus of contemporary Serbian”, INFOtheca,Vol. 12, No. 2, pp. 36-47, 2011.

[3] Korpus savremenog srpskog jezika na Matematičkom fakultetu Univerziteta u Beogradu, Available: http://www.korpus.matf.bg.ac.rs

[4] Ćosić, J., “Primena leksičkih resursa u sentiment analizi teksta”, University of Belgrade Faculty of Mathematics, Serbia, Thesis, 2020.

[5] Antić, P., “Evaluation and improvement of stemmers for Serbian language”, 53rd International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST 2018), Sozopol, Bulgaria, Issue 1, pp. 326-329, June 28-30, 2018.

[6] Kešelj, V., Šipka,D., “A Suffix Subsumption-BasedApproach to Building Stemmers and Lemmatizers for HighlyInflectional Languages with Sparse Resources,” INFOtheca, Vol. 9, No. 1–2, pp. 23a–33a, 2008.

[7] Milošević, N., “Stemmer for Serbian language.” arXiv1209.4471, 2012.

[8] Ljubešić, N., Boras, D., Kubelka, O., “Retrieving Informationin Croatian: Building a Simple and Efficient Rule-Based Stemmer”,Department for Information Sciences, Faculty of Humanities and Social Sciences, Zagreb, Croatia, INFuture2007: Digital Information and Heritage, pp. 313-320, 2007.

[9] Valencia-Zapata, G. A., Mejia, D., Klimeck, G., Zentner, M. G., Ersoy, O., "A Statistical Approach to Increase Classification Accuracy in Supervised Learning Algorithms", IPSI,IPSI Transactions on Internet Research, Vol. 13, No. 2, pp. 27-32. 2017.

[10] Google Developers Portal - Imbalanced data, Accessed: 21.3.2023., Available: https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data

[11] Berthold, M.R. et al., "KNIME: The Konstanz information miner: version 2.0 and beyond",ACM, ACM SIGKDD Explorations Newsletter, Vol. 11, No. 1, pp. 26-31. 2009.

[12] Furnkranz, J., "A study using n-gram features for text categorization.",Austrian Research Institute for Artifical Intelligence, Tehnical report,pp. 1-10. 1998.

[13] Náther, P., "N-gram based Text Categorization", Comenius University, Bratislava, Slovakia,Thesis, 2005.

[14] Cavnar, W. B., Trenkle, J. M., "N-Gram-Based Text Categorization".Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval, Vol. 161175, pp. 161-175. 1994.

[15] Drajić, D.,Ivaniš, P.,"Uvod u teoriju informacija i kodovanje",Akademska misao, Vol. 3, 2009.

[16] Cohen, J., "A coefficient of agreement for nominal scales". Educational and Psychological Measurement, Vol. 20, No.1, pp. 37-46. 1960.