FACTA UNIVERSITATIS (NIŠ)
Ser. Math. Inform. Vol. 34, No 5 (2019), 973–994
https://doi.org/10.22190/FUMI1905973D

THE INFLUENCE OF TEXT PREPROCESSING METHODS AND TOOLS ON CALCULATING TEXT SIMILARITY

Đorđe Petrović and Milena Stanković

Abstract. Text mining to a great extent depends on the various text preprocessing techniques. The preprocessing methods and tools which are used to prepare texts for further mining can be divided into those which are and those which are not language-dependent. The subject matter of this research was the analysis of the influence of these methods and tools on further text mining. We first focused on the analysis of the influence on the reduction of the vector space model for the multidimensional representation of text documents. We then analyzed the influence on calculating text similarity, which is the focus of this research. The conclusion we reached is that the implementation of various text preprocessing methods in the Serbian language, which are used for the reduction of the vector space model for the multidimensional representation of text document, achieves the required results. But, the implementation of various text preprocessing methods specific to the Serbian language for the purpose of calculating text similarity can lead to great differences in the results.
Keywords. Text preprocessing; text mining; text similarity.

1. Introduction

Knowledge discovery through open source text analysis is a field of research which evolved from information retrieval research, and is known as text mining [22]. Most text mining and text retrieval methods require the calculation of text distance or text similarity. The text documents which were the subject matter of this research include higher education institutional statutes from the Republic of Serbia. According to Article 56 of the Law on Higher Education [21], a statute is a basic general act of a higher education institution which regulates the day-to-day running of the institution, its operations and management, as well as any other issues of importance for the activities and work of the higher education institution, in accordance with the law. In addition, Article 6 of this law provides autonomy

Received march 12, 2018; accepted September 16, 2019
2010 *Mathematics Subject Classification.* Primary 68U15; Secondary 68T50, 68P01

to universities and other higher education institutions, which, among other things, includes the right to a statute. This means that all higher education institutions in the Republic of Serbia are required to have this regulatory document and to independently determine its content. In other words, statutes regulate the same issues, but for various institutions, and are created and adopted by various institutions; as such they are suited for the study of text similarity through the implementation of the appropriate methods and tools.

Our focus is the analysis of the influence of text preprocessing methods and tools on text mining, on the reduction of the vector space model for the multidimensional representation of text documents, and on calculating text similarity. The procedure we implemented in this research consists of the following steps:

1. The collection of the text documents required for the research (described in Section 3)
2. Text preparation – preprocessing (described in Section 4)
3. The vector space model for the multidimensional representation of text documents (described in Section 5)
4. The analysis of the influence of text preprocessing methods and tools on the size of the vector space model for the multidimensional representation of text documents (described in Section 6)
5. Calculating text similarity (described in Section 7)
6. The analysis of the influence of text preprocessing methods and tools on calculating text similarity (described in Section 8).

The aim of this research was to indicate the possible problems which could emerge as a result of careless application of certain preprocessing tools on texts written in the Serbian language, carried out for purposes of machine learning. The findings should facilitate the improvement in the aforementioned tools, with the aim of improving the results obtained from texts written in the Serbian language by applying machine learning tools and techniques.

2. Related work

The idiosyncrasies of the research into the Serbian language were studied by [23], while the conclusion that was reached by [10] is that natural language processing of the Serbian language represents quite a challenge.There are several articles related to text preprocessing methods and tools for text mining in the Serbian language, which are also the subject matter of this research. In their work [12] presented a general suffix method for the construction of stemmers and lemmatizers for languages with numerous inflectional suffixes but with limited resources, such as the Serbian language. The evaluation, which was carried out on valid data, provided their method

with an accuracy of 79%. In his work, [16] presented a stemmer for suffix removal in texts written in the Serbian language. This stemmer provided the stimulus for further research in the field of texts written in the Serbian language. The effects of morphological normalization and word embeddings on sentiment classification in documents written in the Serbian language were studied by [5]. They evaluated the influence of lemmatizers and stemmers on classifiers, which were trained and evaluated on a subgroup of data consisting of movie reviews written in Serbian. The authors determined that the use of stemmers was better than the use of lemmatizers, in the specific conditions of their research, both in terms of the accuracy of classification, but also in terms of the effectiveness of normalization. However, unlike the results which they obtained, according to which a stemmer [14] proved to be the best candidate in using n-gram functions of a higher order, in our research, carried out for the purpose of determining similarity between longer texts, we obtained results which are contrary to theirs. They all provided their own significant contribution to furthering research in the field of natural language processing of the Serbian language. Due to its similarity with the Serbian language, the implementations that were developed for the Croatian language could also prove to be of some use [14]. The aim of our research was to analyze the text preprocessing methods and tools used for texts written in the Serbian language on a sample collection of text documents which are the focus of research, and to provide our own contribution to this field.

With the exception of the papers which describe research focused on the Serbian language, we would also like to point out the following works which included other languages. In an experiment which was carried out on a corpus of political documents, written in Czech, the authors [7] studied the influence of text preprocessing techniques for the purpose of uncovering plagiarism. In their study, they did not rely on stemming, but instead on lemmatization, as one of the text preprocessing procedures for further research. In another work, [2] dealt with the categorization of texts written in Arabic, and carried out research on the influence of text preprocessing methods on the performance of three machine learning algorithms, Naive Bayesian, DMNBtext and C4.5. Unlike the aforementioned papers, this research, among other things, relied on four different tools for stemming, and evaluated their influence on the calculation of similarities among the studied texts.

3. Collecting text documents

According to the Regulations on Standards and Procedures for the Accreditation of Higher Education Institutions [20], a special standard was prescribed which refers to "Administrative transparency". This standard indicates that a higher education institution should have its own website and that it should publish various information and documents from its domain. As a result, the process of gaining insight into higher education institutional statutes in Serbia has considerably been facilitated, and for the purpose of this research we have collected these statutes.

The Commission for Accreditation and Quality Assurance [11], in its "Guide for

students", regularly updates the information on accredited study programs in the Republic of Serbia. In addition to the accredited study programs, this guide also includes the basic information on accredited higher education institutions.

From that document we collected information on all the accredited higher education institutions, including their website addresses, and from these addresses during the spring of 2018 we collected 180 full-text statutes. It is worth mentioning that the number of higher education institutions in Serbia which were accredited at the time when we were collecting these texts was 222. The difference in the number of institutions and the number of collected full-text statutes is the result of the following: there is a certain number of accredited higher education institutions which do not have the status of a legal entity and as such do not have their own statutes, and several institutions had not posted their statutes on their websites. Furthermore, in certain cases the texts of the statutes which were published on the internet were access-right protected PDF documents and as such could not be used in this research.

4. Text preparation necessary for calculating text similarity – Text preprocessing

Effective text mining is based on sophisticated methods of text preprocessing. Actually, text mining is so dependent on the various preprocessing techniques that we could say that to a certain extent it is defined by these detailed preparations [8]. The character set in a particular text document as a rule consists of expressions which are made up of various concepts from the vocabulary of a particular language, in which case the vocabulary represents a database consisting of all the words used in that language. These expressions are usually the result of the use of multiple tense variations and other grammatical changes made to words. In addition, within texts we can often find words which have the same form, but different meanings (homonyms), words with different forms but similar meanings (synonyms), words written in capital letters, etc. All of these occurrences should be processed when a text is being converted into basic concepts or terms used to determine term frequency. As a result, preprocessing is aimed at creating a sparse, multidimensional representation of text documents [1]. The concept of "sparseness" refers to the dimensionality of the vectors, since the number of words and phrases that appear in the texts is far smaller than the number of occurrence of words and phrases in the dictionary.

Most of the text preprocessing procedures include a normalization procedure [17]. Text normalization in the Serbian language, due to the idiosyncrasies of the language itself [23], represents a true challenge [10]. For the purpose of this research, text preprocessing included the following steps:

- Raw Text Extraction
- Removing hyphens and other punctuation marks

- Tokenization – the segmentation of the text into its component parts
- Case folding
- Elimination of stop-words
- Stemming.

4.1. Raw text extraction

The statutes we collected were in the form of text documents. The first step in text preprocessing for the purpose of text mining was the conversion of raw text into a character sequence. The textual representation of a language is a character sequence, but text is frequently found in binary format, such as Microsoft Word document format or Portable Document Format (PDF). Since the format of higher education institutional statutes is not prescribed in the Republic of Serbia, it was left to the institutions themselves to make these decisions. Based on the text documents we collected from the Internet, 94,4% of the statutes were published in PDF format, while 5.6% of the statutes were published in *.DOC or *.DOCX format. This means that it was necessary for the collected documents, which represent a group of bites, to be converted into a character sequence.

According to [1], the factors which can influence the conversion of binary documents into text are the following:

1. Certain text documents can be represented in a certain type of code, depending on document format, such as Microsoft Word documents, PDF documents or ZIP files;
2. The language of the document defines its coded character set.

Certain coding systems are very sensitive to the character set being used, and not all coding systems can be used to manipulate all character sets with equal success. For a document which is written in one language, for example Serbian, a different character set is used than for a document written in another language, such as English. For the English language, as for many other European languages, the Latin character set is used, for which we use the abbreviation ASCII character set. The Unicode Consortium designed a standard character set known as the Unicode, in which each character is represented by a unique identifier [1]. What is more, almost all the symbols contained in various languages (including mathematical symbols and numerous ancient characters) could be presented using Unicode. This is another reason why Unicode has become the standard for representing all languages. UTF-8 is often used in various systems, and is especially suited to the ASCII character set. However, even though it is possible to use UTF-8 to code practically any language (which represents the dominant standard), many languages are presented in other codes as well. The type of code which is being used depends on the language, the individual who created the document and the platform on which the document

can be found. In numerous cases, the meta-data of the document contains useful information on the type of coding, rendering any analysis of the document content unnecessary. The collection of documents which are the subject matter of this research did not contain any meta-data on the used character set, but since the texts analyzed in this research were written in Serbian, this means that the UTF-8 character set was used to create these texts.

For the purpose of this research, we preprocessed the texts for further analysis by converting them into plain text. This enabled us to overlook any coding, formatting and text editing. The obtained collection of documents was in the *.TXT format, where the text of one of the statutes was saved in document form using the UTF-8 character set.

4.2. Hyphens and other punctuation marks

During text preprocessing, the work on punctuation marks and hyphens in particular, and their role in the text, requires special attention. In some cases these marks define the borders between words, an in other cases they are a component part of the expression, which should be studied as a single unit [1]. Furthermore, we should also make allowances for the possibilities that during text-wrapping hyphens can be used to divide words into syllables, in such cases when one part of a word is located in one line of the text, while the other part of the word is located in another line. In this case, segmentation would lead to a change in semantic meaning.

Depending on the type of research and the need to incorporate semantic meaning, multi-word expressions and hyphenated expressions, or expressions with other punctuation marks can, but need not necessarily, be divided into separate words, for the purpose of text mining. Since the subject matter of this research is not only semantic meaning, nor have the analyzed texts been divided into syllables, we have omitted all the punctuation marks from the texts, which means that all the multi-word expressions which contained these punctuation marks were divided into separate words. Thus the number of characters in the texts was, on average, reduced by 4.2%. (The average number of characters in the texts was 110692.7, and following the removal of punctuation marks, the average number of the characters was 106051.5).

4.3. Tokenization

Prior to any kind of further text processing, a continuous set of characters was needed for the division of the text into its component parts. Documents can be divided into chapters, sections, paragraphs, sentences, words, and even syllables or letters. The approach most often adopted in text mining is tokenization [8]. A token is a continuous character sequence with a semantic meaning, with the addition that tokens can be repeated with no additional processing (such as stemming) [1]. The process by means of which the text is segmented into its component parts or tokens,

most of which correspond to the words of the language the text was written, is known as tokenization [8]. Solving challenging problems from the perspective of deciding on where the borders between the words in a text are located is an essential part of this process [1]. There is no unique way of performing tokenization. When a text is being read by people, they "perform tokenization" accurately and without giving it much thought, but it turns out that this task is much more ambiguous during natural language processing. What this in fact means is that various tokenization programs create somewhat different segmentations, and as a result the main rule which should be adhered to is that the same tokenization applications must be used consistently on all the texts in a particular research [1]. For the purpose of tokenization in this research, we relied on the open source library "Natural Language Toolkit" (NLTK) [6], so that once the tokens were extracted from the collection of documents, they could further be transformed in order for us to finally obtain the basic concepts and the needed term frequency.

4.4. Case folding

Using upper-case and lower-case letters often defines the semantic interpretation, which is relevant for text mining [1]. Capital letters are used in texts for a variety of reasons, such as the beginning a sentence, or as part of a heading, or because proper nouns are being used. The entire process of case folding is also known as "truecasing" [13]. However, there are limitations to the application of case folding. One of the limitations is ambiguity, which can occur in cases when letters of different sizes provide various meanings of the same words, and there are other examples as well. However, in many cases it is possible to use this procedure. Even though it is not perfect, the simplicity of its application enables efficiency in text preprocessing and editing. In our research, we used an algorithm for case folding, which transformed the entire text into lower-case letters, so as to avoid the problem of various interpretations of the same segment of text, if it is written in a combination of upper-case and lower-case letters.

4.5. Stop-word removal

Stop-words are usual words found in any language, but they do not contribute to the variety of the content [1]. For example, when it comes to classifying articles based on field of study, they represent words with approximately similar term frequencies, for example in articles on sport, or in articles on politics. As a result, it is logical to remove such words which have a negative effect on determining the differences between texts. It is customary to use the following strategies for removing such words [1]:

1. All preposition, adverbs and conjunctions are stop-words. Sometimes pronouns are treated as stop-words as well;

2. There are vocabularies to be found listing the stop-words of a particular language;

3. It is possible to identify frequent tokens in a particular collocation, and to set the boundary at a particular frequency in order to remove any stop-words.

The removal of stop-words is a hard variant of a softer approach of down-weighting frequently used words with inverse document frequency normalization. In some cases there is loss of information connected to the hard removal of stop-words. As a result, many search systems and mining systems do not remove stop-words, and instead rely on an approach for reducing the weight of frequent stop-words [1].

For the purpose of this research, we have devised our own collection, which consists of 288 words without any significant content value, written in the Cyrillic alphabet, and these words were removed from the texts for the purpose of text mining.

4.6. Usage-based consolidation

The concept of usage-based consolidation is a procedure similar to stemming, except that it is a simpler process and is applied during tokenization with the use of the lookup tables [1]. The basic idea is that small variations of the same token often refer to the same word. These variations usually occur depending on the writer, the author of the text, the geographic area from which the text originated, the use of dialects, accents, and the like. In all such cases it is important that these variations in the text are consolidated into a single term. For example, we might in practice make and use tables or other data structure techniques on all the possible variations of tokens with their standardized forms. Bearing in mind that the subject matter of this research is a collection of higher education institutional statutes, this means that the texts included in these documents are formal, without any of the previously cited variations, and as a result it was not necessary to apply usage-based consolidation on this group of documents.

4.7. Stemming

Stemming is a process of consolidating related words which have the same root [1], and in any text mining process, stemming requires special attention. It is a process whose application depends solely on the language in which the analyzed texts were written, and thus, the influence of this process on text mining is language-specific. For example, text documents could contain one and the same word in either singular or plural form at various times, or some other variations of that same word. In such cases, it makes sense for all those variations to be consolidated into a single word, since changes made to the word do not alter its sematic meaning from the point of view of text mining.

Stemming usually refers to the process of extraction of the morphological root of the word, while various heuristics are used to achieve that goal. The usual techniques include [1]:

1. Semi-automatic lookup tables, created up front in a semi-automatic manner, with various heuristics;
2. Suffix stripping, a list of stored rules used to find the root form of a particular word. These rules can also be used to strip prefixes, although it is more common for them to be used to strip suffixes;
3. Lemmatization – a sophisticated approach which uses a certain part of speech to determine the root of a word. The rules of normalization depend on the part of speech and are language-specific to a great extent.

Sometimes it is considered that lemmatization differs from stemming because it exceeds the simple rules of stripping and relies on the morphological roots of the word. This approach provides a dictionary form of the given word which is known as a lemma. A lemmatizer requires an extensive vocabulary to perform its tasks, along with language-specific knowledge, compared to the other stemming tools. The classic lemmatization algorithm is Porter's algorithm [18]. The most recent version of Porter's algorithm is also known as "Snowball".

Several previously published implementations of stemming algorithms for the Serbian language already exist. In their study, the authors [4] relied on the following implementations:

- A stemmer and lemmatizer for resource-limited languages with extensive inflectional affixes, based on the inclusion of suffixes [9]
- A stemmer for the Serbian language [16]
- A stemmer for the Croatian language [14].

Based on the research carried out by [12], the stemmer accuracy for the Serbian language which they tested was somewhat less than 80%. It is worth mentioning that the final cited stemmer was designed for use on texts written in the Croatian language, and due to the similarities between Serbian and Croatian, it is possible to use this stemmer, under the appropriate conditions, for texts written in Serbian. The implementation of this stemmer for texts written in Serbian has even received excellent reviews in a study published by [9].

For the previously cited stemmers, the expectation is that the input text will be formatted by using the UTF-8 character set, and that the output text will be obtained in this character set. Since two alphabets are used in this language, the Cyrillic and Latin, the input texts for the Serbian stemmers can be in either alphabet, but the output texts the stemmers produce are always in the Latin alphabet.

The stemmers for Serbian language internally use the so-called dual1 coding system which allows only Latin script, without any diacritical marks. In order to generate texts in this coding system, all the letters of the Cyrillic alphabet are first converted into their Latin counterparts, and after that all the letters with diacritical marks are replaced by their appropriate counterparts without these marks (for example, Č/č is replaced by Cx/cx, Ć/ć by Cy/cy, Dž/dž by Dx/dx, Đ/đ by Dy/dy, Ž/ž by Zx/zx, Š/š by Sx/sx, Lj/lj by Ly/ly and Nj/nj by Ny/ny).

In our research, for the purpose of stemming, we relied on the open source implementation of a collection of stemmers for the Serbian and Croatian language, the work of Vuk Batanović [3]. The delivered library, written in the Java program language, among other things enables the processing of content of textual data files from the command line, using the following command:

```
java -jar SCStemmers.jar StemmerID InputFile OutputFile
```

where the StemmerID is the number of the appropriate algorithm:

1. Kešelj & Šipka – Greedy [12], herein "Stemmer1"
2. Kešelj & Šipka – Optimal [12], herein "Stemmer2"
3. Milošević [16], herein "Stemmer3"
4. Ljubešić & Pandžić [14], herein "Stemmer4".

5. A vector space model for the representation of text documents

The vector space model for the multidimensional representation of text documents is used in most research applications [1]. This vector space model contains one dimension for each word, and the value of the dimension is always positive when the word is present in the text. If the situation is opposite, the value is zero. The positive value can be the normalized term frequency of that word, or the binary value indicator 1. In an analyzed document, the number of words which the document contains is a small sub-sample of the vocabulary of a particular language. It is not unusual for a collection of documents, which make up a vocabulary, to contain upwards of one hundred thousand words, while the average number of words in a document can be as small as several hundred.

We should bear in mind that the conversion of a document into a representation in the vector space model includes a loss of information on the word order in the text document. As a result, this model is referred to as the bag-of-words model [1]. Two of the most frequently used multidimensional representations of textual data, which are suited to the binary model are the "Bernoulli" or "boolen" model) and the "tf-idf" model.

In some applications it is sufficient to use a binary 0-1 representation, which provides information on whether a word is found in the document or whether it is not. However, through binary representations, a lot of information is lost, since

this type of representation does not contain the term frequency of individual words in the text, and there is also no normalization which could help determine the importance of the words in the text [1]. However, the main advantage of the binary representation is the compactness and possible use of applications, including those which deal with the presence or absence of certain words in a document.

However, most of the representations of text documents do not function with a binary model, and instead rely on normalized term frequency. This model is referred to as the "tf-idf" model, where "tf" refers to the frequency or number of occurrences of a term in a document, while "idf" refers to the inverse frequency, that is, the number of documents which contain a particular term. If $\overline{X} = (x_1 \ldots x_d)$, where d is the dimensional representation of a particular text document, following the extraction phase of a term, then x_i represents the number of repetitions of a term in a document. Since word frequency in long documents can sometimes differ significantly, there is reason to resort to dumping functions for these frequencies. A square root function or logarithm could be used to decrease the effect of spam [1]. The inverse document frequency id_i, of an i^{th} term is the number of documents in which this term is found: $id_i = log(n/n_i)$ [1]. Term frequency normalization is normalized by multiplying the number of occurrences of a term with the inverse document frequency: $x_i = x_i * id_i$.

6. The analysis of the influence of text preprocessing methods and tools on the size of the vector space model for the multidimensional representation of text documents

The number of characters in the analyzed collection of documents, prior to and following the removal of punctuation marks is given in the following table:

Table 1: An overview of the number of characters in the analyzed collection of documents

Text ID:	**1**	**2**	**3**	**...**	**Average number of characters in the analyzed texts:**
Raw text, without any kind of preprocessing, the number of characters in the text:	37904	220283	160648	...	**110692.7**
Text with the removed punctuation marks, the number of characters in the text:	37010	201385	155420	...	**106051.5**

Based on the information shown in the previous table, it is clear that following

the removal of punctuation marks, the average number of characters is reduced by approximately 4.2%.

The number of tokens in the analyzed collection of documents, following the removal of punctuation marks, and according to the subsequent steps of text preprocessing, is provided in the following table:

Table 2: An overview of the number of tokens in the analyzed collection of documents, based on the steps of text preprocessing

Text ID:	**1**	**2**	**3**	**...**	**The average number of characters in the analyzed texts:**
Number of tokens in the text:	5548	30039	22595	...	**15104.1**
Number of unique tokens in the text:	881	2040	1867	...	**1620.0**
Number of unique tokens following case folding:	761	1813	1653	...	**1439.1**
Number of unique tokens following the removal of stop-words:	734	1786	1622	...	**1414.1**

Based on the presented data, it is clear that each step in its own way contributes to the reduction of the vector space model for the multidimensional representation of text documents, and as a result further processing will be more efficient.

The following step in text preprocessing was stemming. In this step four different algorithms were used, for the same purpose, each of which in its own way contributed to a further reduction in the vector space model for the multidimensional representation of text documents. The following table contains data on the number of unique tokens following the application of each of these algorithms:

Table 3: The number of tokens in the analyzed collection of documents, following the implementation of various stemming algorithms

Text ID:	**1**	**2**	**3**	**...**	**Average number of tokens in the analyzed texts:**
Number of unique tokens following the implementation of the Stemmer1 algorithm:	428	945	858	...	**775.9**
Number of unique tokens following the implementation of the Stemmer2 algorithm:	407	907	814	...	**736.3**
Number of unique tokens following the implementation of the Stemmer3 algorithm:	458	970	892	...	**801.9**
Number of unique tokens following the implementation of the Stemmer4 algorithm:	734	1786	1622	...	**1388.9**

7. Calculating text similarity

Numerous applications for multidimensional data mining rely on the Euclidean distance in order to calculate the distance between pairs of points. At first glance, it would seem that for the calculation of the distance between texts, a text needs to be viewed as a special case of multidimensional data. However, the Euclidean distance is not suited for calculating the distance in multidimensional representations which are very sparse, and in which the number of zero values significantly varies in various points. And this is precisely what often occurs in texts, as a result of various document length. Euclidean distance will consistently report higher values for the distance between pairs of documents, even if large fractions of these documents are common [1]. This indicates that the use of functions which significantly normalize distance values (or similarity values) between texts, for texts of various lengths, is needed. The natural solution to this problem is the use of the cosine between multidimensional vectors which represent two documents [1]. This is because the cosine between a pair of vectors does not depend on vector length, but only the angle between them. Besides, cosine similarity is suitable for cases in which term frequency is explicitly stated [1]. This normalization also ensures that the cosine value always ranges between 0 and 1. In other words, the cosine of the angle is a geometric average of the fraction with segmented words which are contained in each pair of documents (in case a binary representation of a text document is being used). Even in the case when the values of *tf-idf* are used instead of binary values,

this factor plays a dominant role in calculating the cosine. Since the calculation of the cosine to a great extent depends on the fractions of general words in each of the analyzed documents, this function does not to a great extent depend on the length of the document.

Upon completion of text preprocessing, as was previously explained, the prepared documents were converted into vector form following the "tf-idf" model. In our research, as a measure of text similarity, we calculated the cosine of the angle between pairs of vectors. Thus we obtained a similarity matrix S, which contains obtained data on text similarity:

$$S = |s_{ij}| = \begin{vmatrix} s_{11} & s_{12} & \dots & s_{1n} \\ s_{21} & \dots & \dots & \dots \\ \dots & \dots & \dots & s_{(n-1)n} \\ s_{n1} & \dots & s_{n((n-1)} & s_{nn} \end{vmatrix}$$

where

S – is the similarity matrix for the text documents

s_{ij} – is the similarity between texts i and j, of the analyzed group of documents.

It is clear that this is a similarity square matrix, for which the following holds:

- $s_{\mathrm{ii}} = 1$, for each i
- $s_{\mathrm{ij}} = s_{\mathrm{ji}}$, for each i and for each j

$$S = |s_{ij}| = \begin{vmatrix} 1 & s_{12} & \dots & s_{1n} \\ s_{21} & 1 & \dots & \dots \\ \dots & \dots & 1 & s_{(n-1)n} \\ s_{n1} & \dots & s_{n((n-1)} & 1 \end{vmatrix}$$

Seeing how we used four different stemming algorithms for the Serbian language in this research, we obtained four matrices with values for text similarity:

- S_1 – the similarity matrix for the analyzed documents following the implementation of the Stemmer1 algorithm,
- S_2 - the similarity matrix for the analyzed documents following the implementation of the Stemmer2 algorithm,
- S_3 - the similarity matrix for the analyzed documents following the implementation of the Stemmer3 algorithm,
- S_4 - the similarity matrix for the analyzed documents following the implementation of the Stemmer4 algorithm.

An overview of the obtained values of these matrices is:

$$S_1 = |s_{ij}|_1 = \begin{vmatrix} 1 & 0.606 & \dots & 0.672 \\ 0.606 & 1 & \dots & \dots \\ \dots & \dots & 1 & 0.900 \\ 0.672 & \dots & 0.900 & 1 \end{vmatrix},$$

$$S_2 = |s_{ij}|_2 = \begin{vmatrix} 1 & 0.621 & \dots & 0.658 \\ 0.621 & 1 & \dots & \dots \\ \dots & \dots & 1 & 0.907 \\ 0.658 & \dots & 0.907 & 1 \end{vmatrix},$$

$$S_3 = |s_{ij}|_3 = \begin{vmatrix} 1 & 0.626 & \dots & 0.652 \\ 0.626 & 1 & \dots & \dots \\ \dots & \dots & 1 & 0.909 \\ 0.652 & \dots & 0.909 & 1 \end{vmatrix},$$

$$S_4 = |s_{ij}|_4 = \begin{vmatrix} 1 & 0.565 & \dots & 0.631 \\ 0.565 & 1 & \dots & \dots \\ \dots & \dots & 1 & 0.869 \\ 0.631 & \dots & 0.869 & 1 \end{vmatrix}.$$

As the final result for text similarity for the analyzed group of documents, we used the average similarity value, obtained through the implementation of the cited stemming algorithms:

$$\overline{s_{ij}} = \frac{\sum (s_{ij})}{n} = \frac{\sum (s_{ij})}{4}$$

where

$\overline{s_{ij}}$ – is the average value for text similarity, taking into consideration all of the used stemming algorithms

$n = 4$ – is the number of the various algorithms used in this research.

An overview of the average values for text similarity is provided in the following matrix:

$$\overline{S} = |\overline{s_{ij}}| = \begin{vmatrix} 1 & 0.605 & \dots & 0.653 \\ 0.605 & 1 & \dots & \dots \\ \dots & \dots & 1 & 0.896 \\ 0.653 & \dots & 0.896 & 1 \end{vmatrix}$$

The following table provides an overview of the values for similarities between individual texts following the implementation of various stemming algorithms, as well as the calculated average values for text similarity:

Table 4: The similarity between Text(i) and Text(j) following the implementation of a stemming algorithm

i	**j**	**Stemmer1**	**Stemmer2**	**Stemmer3**	**Stemmer4**	**Average similarity**
1	1	100.0%	100.0%	100.0%	100.0%	100.0%
1	2	60.6%	62.1%	62.6%	56.5%	60.5%
1	3	77.7%	77.7%	78.5%	74.3%	77.1%
1	4	78.3%	78.6%	79.1%	75.5%	77.9%
...	...	...	...	...	...	...

8. The analysis of the influence of text processing methods and tools on calculating text similarity

In this section, we will focus on the following analyses:

- The analysis of the influence of the overall procedure of text processing on calculating text similarity
- The analysis of the influence of various stemming algorithms and tools for the Serbian language on calculating text similarity.

8.1. The influence of text preprocessing on calculating text similarity

We compared the calculated values for text similarity, obtained following the implementation of the overall text preprocessing procedure given in Table 4, with the values which would have been obtained if the similarities between the texts had been calculated without the text preprocessing procedure. The following table provides an overview of these values for the individual texts:

Table 5 Similarities between the analyzed texts following text preprocessing and without text preprocessing

i	**j**	**Following text preprocessing**	**Without text preprocessing**
1	1	100.0%	100.0%
1	2	60.5%	65.6%
1	3	77.1%	91.9%
1	4	77.9%	91.8%
...	...	...	...
		Average value: 71.3%	**Average value:** 80.5%

The average value of text similarity which we calculated taking into consideration all the implemented stemming algorithms is $\overline{s_{ij}} = 71.3\%$. The average value for text similarity in raw form, without any text preprocessing procedures, is $\overline{s_{ij}} = 80.5\%$. Based on these values, we can conclude that the text preprocessing procedures led to a decrease in the values of similarity, compared to the values which would have been obtained if we had calculated text similarity without preprocessing. In other words, the differences between the texts are evident when text similarity is calculated following text preprocessing.

8.2. The influence of the stemming algorithm on calculating text similarity

The following table presents the average values for text similarity following the application of various stemming algorithms, calculated for various texts (indexes i and j differ one from the other):

Table 6 Average values for text similarity following the implementation of various stemming algorithms

Stemming algorithm ID	**Stemmer1**	**Stemmer2**	**Stemmer3**	**Stemmer4**
Average values for text similarity	73.2%	71.9%	73.2%	66.9%

Based on the obtained data, it can be concluded that the various stemming algorithms contributed to the difference between the obtained similarity results. We can note that from the standpoint of average values, similar results were obtained following the implementation of the "Stemmer1" and "Stemmer3" algorithm.

Viewed from the standpoint of similarities between individual pairs of texts, from the standpoint of results obtained through the implementation of a stemming algorithm, we looked for documents for which the similarity values calculated based on individual algorithms differ the most from the average values. The following table presents the calculated text similarity for the previous examples, as well as the calculated average similarity values.

Table 7: Individual pairs of texts for which the calculated similarity differs the most from the average values

i	**j**	**Stemmer1**	**Stemmer2**	**Stemmer3**	**Stemmer4**	**Average similarity**
...	...	...	...	...	...	...
16	116	55.0%	58.8%	**66.2% (a difference of 20.4%)**	3.1%	45.8%
...	...	...	...	...	...	...
75	116	75.7%	**77.7% (a difference of 18%)**	79.8%	5.7% **(a difference of 54%)**	59.7%
...	...	...	...	...	...	...
193	196	**68.9% (a difference of 16.6%)**	69.6%	68.1%	2.7	52.3%
...	...	...	...	...	...	...

The data from the previous table provide the following information:

- following the implementation of the "Stemmer1" algorithm, the greatest deviation of this algorithm from the average value is 16.6%. For text documents with the ID 193 and 196, the implementation of this algorithm yielded a similarity of 68.9%, while the average value of the similarity for this pair of documents was 52.3%, taking into consideration all the algorithms.

- following the implementation of the "Stemmer2" algorithm, the greatest deviation of this algorithm from the average value is 18%. For text documents with the ID 75 and 116, the implementation of this algorithm yielded a similarity of 77.7%, while the average value of the similarity for this pair of documents was 59.7%, taking into consideration all the algorithms.

- following the implementation of the "Stemmer3" algorithm, the greatest deviation of this algorithm from the average value is 20.4%. For the text documents with the ID 16 and 116, the implementation of this algorithm yielded a similarity of 66.2%, while the average value of the similarity for this pair of documents was 45.8%, taking into consideration all the algorithms.

- following the application of the "Stemmer4" algorithm, the greatest deviation of this algorithm from the average value is as much as 54%. For the text documents with the ID 75 and 116, the implementation of this algorithm yielded a similarity of 5.7%, while the average value of the similarity for this pair of documents was 59.7%, taking into consideration all the algorithms.

By analyzing the data presented in the previous table, but also the sample texts which were the subject matter of this research, the results obtained by implementing the "Stemmer4" algorithm have indicated great deviations from the average value. If we take into consideration the fact that in this research the results of the implementation of this algorithm had an impact on the calculation of average values, the difference between the results obtained through the implementation of this algorithm and of the other algorithms is even greater. Thus, for example, for the text documents with the ID 75 and 116, the difference between the calculated values of similarity through the implementation of this algorithm and the other algorithms exceeds 70%. Based on that, and the sample of texts which were the subject matter of this research, we could conclude that the implementation of stemmers designed for use for other, foreign languages, in comparison to other algorithms, leads to great differences when calculating text similarity, and that as such they cannot be applied to the Serbian language despite the great similarity between the languages.

The textual data which were used in this research are publicly available, and belong to a group of formal, legal texts, written by various authors. As such, they as classified as "long" texts and as a result, "short" texts such as comments or online tweets were not the subject matter of this research. Based on the aforementioned, the data used in the research could in no way influence the research results, that is, this research could be replicated on another group of data.

While the effect of the stemming algorithms is significant for information retrieval applications, the use of this algorithm is not as critical when a large collection of documents of a reasonable length is being analyzed [1]. Techniques like stemming help compensate for data sparseness. Actually, overly aggressive stemming can easily degrade classification performance [15].

9. Conclusion

For any kind of text mining, it is first necessary to implement methods and tools used to prepare these texts for text mining, i.e. text preprocessing is a necessary first step. The methods and tools used for that purpose can be classified into two groups: those which are not dependent on the language the analyzed texts were written in and those which are language-dependent.

The implementation of methods and tools which are not language-specific and do not depend on the language of the analyzed texts is broad. They are mostly used to reduce the vector space model for the multidimensional representation of text documents, and their implementation leads to appropriate results.

Problems might occur in the implementation of text preprocessing methods and tools which are language-specific. For texts written in the Serbian language, based on the information we have at our disposal, there is no open source algorithm or lookup table for token Usage-Based Consolidation. When it comes to the implementation of the stemming process, there are several such implementations, but none of them still provide maximum accuracy, nor are they widely used. Based on

the research we have carried out and described, we have reached the conclusion that the implementation of various text preprocessing methods in the Serbian language, for the purpose of calculating text similarity, may lead to great differences in the results. On the sample of texts which were the subject matter of this research, it could be concluded that the implementation of stemming algorithms, which were designed for use in other languages, could lead to great differences in calculating text similarity.

Based on the aforementioned, and the sample of texts which were the focus of this research, it could be concluded that the implementation of stemming algorithms designed for use in other languages, compared to other algorithms, leads to great differences when calculating text similarity, and as such they cannot be applied in the case of the Serbian language, despite the similarity between the languages.

All these conclusions indicate that care should be taken when implementing various text preprocessing methods and tools.

Future work will include continued research on the tools and methodologies which refer to machine learning from texts, where special attention will be focused on the tools and methodologies which refer to texts written in the Serbian language. It can be noted that, based on the results of this research and also on the results of similar studies, there is a difference in the results obtained following the application of certain preprocessing tools used on texts written in Serbian, if those tools are applied in the domain of specific research. Further research would include achieving a level at which the application of tools and methodologies for working with the Serbian language will lead to uniform results, irrespective of the domain of research, the sources of the data, and other differences.

REFERENCES

1. Aggarwal, C. C.: *Machine Learning for Text.* s.l.:Springer, 2018.
2. Alshammari, R.: *Arabic Text Categorization using Machine Learning. International Journal of Advanced Computer Science and Applications*, 9(3), pp. 226-230, 2018.
3. Batanović, V., Furlan, B. & Nikolić, B.: *A Software System for Determining the Semantic Similarity of Short Texts in Serbian.* Belgrade, 19thTelecommunications Forum (TELFOR) Proceedings of Papers, 2011.
4. Batanović, V., Nikolić, B. & Milosavljević, M.: *Reliable Baselines for Sentiment Analysis in Resource-Limited Languages: The Serbian Movie Review Dataset.* Portorož, Slovenia, Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016), 2016.
5. Batanović, V. & Nikolić, B.: *Sentiment Classification of Documents in Serbian: The Effects of Morphological Normalization and Word Embeddings*, Telfor Journal, 9(2), 2017.
6. Bird, S., Klein, E. & Loper, E.: *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* s.l.:O'Reilly Media, Inc., 2009.

7. Ceska, Z. & Fox, C.: *The Influence of Text Pre-processing on Plagiarism Detection.* Borovets, Bulgaria, International Conference RANLP, 2009.
8. Feldman, R. & Sanger, J.: *The Text Mining Handbook.* s.l.:Cambridge University Press, 2006.
9. Jones, T.: *Serbian Stemmer Analysis.* [Online], 2017., Available at: https://www.mediawiki.org/wiki/User:TJones_(WMF)/Notes/Serbian_Stemmer_Analysis [Accessed October 2018].
10. Kajan, E., Pljasković, A. & Crnišanin, A.: *Normalizacija tekstualnih dokumenata na sprskom jeziku u cilju efikasnijeg pretraživanja u sistemima e-uprave.* Zlatibor, ETRAN, 2012.
11. KAPK: *Commission for accreditation and quality assurance, Guide for students.* [Online], 2018. Available at: http://www.kapk.org [Accessed 2018].
12. Košelj, V. & Šipka, D.: For the greedy and the optimal subsumption-based stemmer for Serbian: A Suffix Subsumption-Based Approach to Building Stemmers and Lemmatizers for Highly Inflectional Languages with Sparse Resources. *Infotheca,* Tom 9(1-2), pp. 23a-33a, 2008.
13. Lita, L. V., Ittycheriah, A., Roukos, S. & Kambhatla, N.: *Truecasing.* Sapporo, Japan, ACL '03 Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, 2003.
14. Ljubešić, N., Boras, D. & Kubelka, O.: *Retrieving Information in Croatian: building a simple and effcient rule-based stemmer.* Zagreb, 1st International Conference The Future of Information Sciences (INFuture), 2007.
15. Manning, C. D., Raghavan, P. & Schütze, H.: *Introduction to Information Retrieval.* s.l.:Cambridge University Press, 2008.
16. Milošević, N.: *Stemmer for Serbian language,* s.l.: arXiv preprint arXiv:1209.4471, 2012.
17. Miner, G. et al.: *Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications.* s.l.:Academic Press, 2012.
18. Porter, M. F.: An algorithm for suffix stripping. *Program,* 14(3), pp. 130-137, 1980.
19. Schütze, H. & Silverstein, C.: *Projections for efficient document clustering.* Philadelphia, Pennsylvania, SIGIR '97 Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval, 1997.
20. Službeni glasnik RS: *Pravilnik o standardima i postupku za akreditaciju visokoškolskih ustanova, Službeni glasnik RS, broj 88/17.* [Online], 2017. Available at: http://www.kapk.org/en/accreditation/ [Accessed 2018]
21. Službeni glasnik RS: *Zakon o visokom obrazovanju, Službeni glasnik Republike Srbije, broj 73/18.* [Online], 2018. Available at: http://www.parlament.gov.rs [Accessed 2018].
22. Stranieri, A. & Zeleznikow, J.: *Knowledge Discovery from Legal Databases.* s.l.:Springer, 2005.
23. Vitas, D. et al.: *The serbian language in the digital age.* s.l.:Springer, Berlin, Heidelberg, 2012.

Đorđe Petrović
Faculty of Electronic Engineering
Department for Computer Science
18000 Niš, Serbia
petrovicdj@gmail.com

Milena Stanković
Faculty of Electronic Engineering
Department for Computer Science
18000 Niš, Serbia
milena.stankovic@elfak.ni.ac.rs