One solution of searching text documents in Serbian language

Adela Crnišanin, Aldina Pljasković, Ulfeta Marovac, Ejub Kajan, *State University of Novi Pazar, Serbia*

A***bstract***— *The modern way of life, e-business, a large amount of data available in electronic form imposed the need for analysis of textual documents written in different natural languages. Natural languages have different grammatical rules and a lot of exceptions, which complicate analyze of documents. In this paper, we propose one solution for searching documents on Serbian language. System components have been described. Also, normalization algorithm, preparing algorithm for grouping similar documents, and searching algorithm have been implemented.*

Key words **— clustering, keyword, n-gram, normalization, similarity searching**

I. INTRODUCTION

E-government may be defined as a way governments use Web-based Internet applications in order to provide more convenient access to government information and services to citizens and businesses, and to improve the quality of services [1]. E-Government deploys four major business models among participating entities: G (Government), C (Citizens), E (Employees), and B (Business). These are G2G, G2C, G2B and G2E. In this work we are concentrated on G2C (Government-to-Citizen) relationship. We are working on developing e-government web application that helps citizens to found answer for their questions. Participants of this project are two city halls and a nation-wide institution. In order to speed up the process of searching for accurate answer (document with relevant content) on new question, the documents must be grouped by their content. First step is automated generation of groups containing keywords, using clustering process. Questions and documents containing the most similar terms will be placed in the same cluster.

When a user puts a new question, it is necessary to calculate its similarity with other questions in the selected cluster. Simplified view of the architecture is depicted in Figure 1.

Citizen's queries are processed by *Query preprocessor* that examines whether the answer alreday exists, and if not, it serach available documents in either local or distributed government databases in a domain, using appropariate wrappers. There are three types of contents in the system: citizen's questions, governmental documents and SME (Subject Matter Expert) answers. Placing document or question in a certain cluster depends on keywords they contain. Fuzzy C - mean algorithm is used for clustering. After examined which cluster new question belongs to, it should be compared with existing questions with the cosine similarity methods between two questions. System is described in detail in paper [2].

In order to perform clustering of documents and measure similarity between sentences, the composition of documents need to be analyzed and, during the analysis, all the peculiarities of the language in which documents are written (in this case, Serbian) need to be applied. Complex Serbian grammar and two official alphabets make searching process for documents in Serbian language real challenge. Ambiguity of words and sentence structures aggravate the problem of analyzing the contents of documents. For analysis purpose, we need different lexical resources: corpus of Serbian language, morphological dictionary, stop-words, dictionary of abbreviations, dictionary of proper names and so on. Difficulties for finding appropriate resources are numerous and depend on the type of document being examined. For example, the meanings of the same words in legal documents and in other documents that describe the natural wealth of a country are different. Additional problems associated with the analysis of text documents in Serbian language are: the lack of adequate resources, constant changes in natural languages (e.g. introduction of new terms) and too much time for processing text when enormous resources are used.

In this paper, we'll show the results for the parts of the system associated with document preparation for clustering process and searching by similarity process. The rest of the paper is organized as follows. The second chapter describes *Query preprocessor* system component. This component prepares document for the next step - *Normalizator* component. The third chapter presents different methods for document normalization. The fourth chapter shows algorithm for extracting keywords from documents and the way documents are clustered. *Advanse clustering* component is responsible for grouping documents by keywords. After that, it is described how *Answering Processor* component works. *Answering Processor* finds response for asked question using searching algorithms based on similarity between two questions or on similarity between question and document. At the end, used technologies have been

described; the significance of the results has been explained and directions for further research presented.

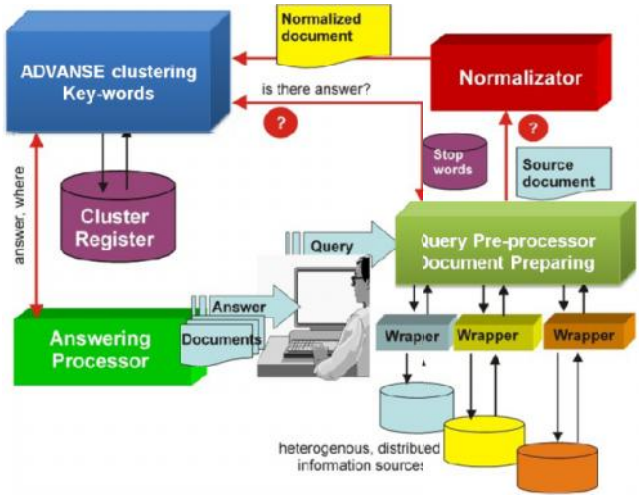

Figure 1. Simplified view to system architecture

II. QUERY PRE-PROCESSOR

Data input into system can be a question or a document. Database consists of huge number of documents. Before being inserted into database, the documents have to be normalized so the database could be optimally exploited for quick and accurate searching for documents in the future. The same procedure is applied when inserting question into database. Words being meaning carriers are determined, while other words are less important and could make searching harder, so they are thrown out. *Query Pre-processor* (Document preparing) component prepares documents and question for normalization process.

Documents contain pure text and are made of sentences. Question can be made of one or more sentences. Therefore, functions related to question preparation are subset of the functions related to the document preparation. Documents for searching purpose are stored in a database and can be TXT, PDF, or HTML format. Depending on document's file format, the different function (Wrapper) for cleaning document of superfluous characters has been used. Other functions are mutual for questions and documents. Therefore, at the following part, the word "document" will be used for both question and document.

Documents often contain redundant signs, or they are written in Cyrillic alphabet or include informal signs. The document must be prepared prior to analysis, in order to be ready for normalization process. At the end of normalization process, all documents, no matter which format they were in, should be back to Latin alphabet, without excessive space between words. The sentence should ends with a period, question mark or exclamation mark. New sentence begins with a capital letter and a space. Document format should be UTF8. Preparation of documents goes through the following three steps:

- Processing documents written in Cyrillic and Latin alphabet
- Processing documents in HTML format and removing html tags
- Remove irregular signs (emoticons, slang,..)
- Remove stop words
- Transferring text from ASCII to UTF8 format and Latin alphabet. This document was prepared for the normalization of the word. (Figure 2)

The resources that are available are following:

•Stop words (adverbs, prepositions, conjunctions, exclamations, words, pronouns)
• Extensions (grammatical suffixes in the form of declension noun and adjective, and verb conjugation[3])
• The terms of informal communication (emoticons, slang).

```
procedure PREPAREDOCUMENT(DocumentId)
ConectToDatabase(Project)
Document = GetDocumentFromDatabase(DocumentId)
StopWords = GetStopWordsFromDatabase()
CyrillicToAscii(Document)
LatinToAscii(Document)
HtmlToText(Document)
DeleteStopWords(Document)
DeleteIrregularSigns(Document)
AsciiToLatin(Document)
Sentences = SplitDocumentOnSentences(Document)
```

Figure 2. Pseudo code algorithm for document preparation

III. *NORMALIZATOR* COMPONENT

Document prepared in *Query Pre-Processor* component need to be further normalized. This step is performed by *Normalizator* component. Reasons for normalizing document are clearly described at the beginning of this paper.

In the paper introduction, it was pointed out that is possible to get more effective search results if documents are grouped according to certain criteria. Documents need to be grouped by the main carriers of their meaning, i.e. keywords.

As the extraction of keywords is time consuming job, any pre-treatment which removes unnecessary words or parts of words, makes extraction faster and more efficient. When the document was prepared for analysis, it is necessary to ensure that the case/verb suffix forms of the same word doesn't appear in keywords, but to ensure different forms is treated as one word. The similarity between two words is determined by their semantic meaning. The root of word is the carrier of the word's meaning. The derivations of compound words in

Serbian language (the existence of a prefix, infix, and suffix) makes difficult to identify words that have a common root. Therefore, prior to keywords extraction, the document goes through another step, which is called the normalization of the word.

In broad terms, the normalization of the text includes the transformation of the text into another form that is suitable for any type of computer processing. In our case, that is searching. The purpose of the normalization of the words is the release of excessive modification of words that do not make changes in their meaning, and the reduction of these modifications on the common, basic form.

There are several ways to normalize words, and they are further described.

Normalization 1

For this type of normalization, it is necessary to have a morphological dictionary. Using this resource, lemmatization of words takes job – removing word form suffixes and word formation suffixes. The advantage of this normalization is the accuracy of the results, but it's achieved with great effort. By taking into account the vocal changes and the scope of resources used, result retrieval in slowed down. Another negative feature of this normalization is that it is specific to one language, and that it is difficult to reach such resources.

Normalization 2

Word's form suffixes are necessary resources for this type of normalization. It is described in detail in [4]. Its disadvantages are encountered in complex words derivation. Since the number of word's form extensions is much smaller than the number of words in the morphological dictionary, the complexity of this algorithm is much smaller, which makes this algorithm much faster than the previous one.

Normalization 3

This is the simplest algorithm, the fastest, which does not require any additional resources. This normalization extracts first *k* letters of the word. Therefore, other words whose length is less than *k* are not considered. Again, the problem of prefixes and other grammatical peculiarities remains unresolved.

Normalization 4 - n-gram word analyze

Unlike previous approaches, this type of normalization solves the problem of prefixes. This is one of the simplest algorithms, but its application significantly increases the number of normalized words.

N-gram is a subsequence consisting of *n* elements of an array. In computational linguistics, N-gram models are commonly used to predict words or letters in applications for different purposes. [5] For example, the word "učiti" is composed of the following ngrams u-č-i-t-i (length 1), uč-či-it-ti (length 2), uči-čit-iti (length 3), učit-čiti (length 4) and učiti (length 5).

N-gram analysis is a procedure applied to the text, and its result is to obtain a set of n-grams of a certain length. N-grams are obtained by moving the frame of length *n*, whose origin may be in positions *1* to m-n +1, where *m* is the length of the string (Figure 3).

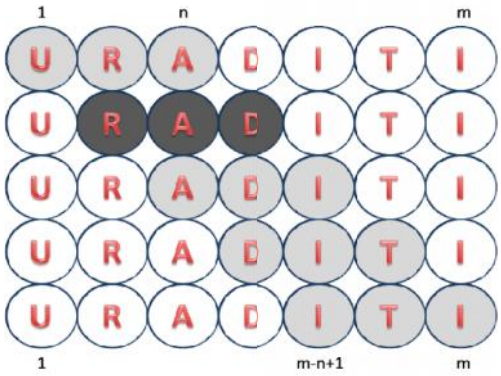

Figure 3. Extracting n-grams from words

N-gram analysis can be used as an alternative to find words with the same root. Thus, the seemingly different computer words, "uraditi", "odraditi", "radi", have a common 3-gram "rad", and semantically are all related to the verb "raditi" they are made of.

In order to obtain efficient results in the selection of *n*, form suffixes for important words the in Serbian language were collected. It was found that the highest percentage (92.70%) suffixes have less than 4 letters. [6] Therefore, this paper deals with the 4-gram analysis.

N-gram analysis was applied over the words, not over sentences, so that the gaps cannot be founded within a single n-gram. This would increase the number of data in the database, and therefore the processing time required. This type of n-gram analysis could be meaningful in some languages, for finding a common phrases or syntagms. This problem was solved in another way, by calculating the matrix of mutual words frequentation, which is described in the following section.

IV. ADVANSE CLUSTERING-KEYWORD EXTRACTION

After normalization, question or document need to be placed into an appropriate question/document group called cluster. The cluster consists of semantically similar documents. Two documents are similar if have mutual keywords. Determining which words are keywords and grouping documents into clusters is task job for ADVANSE component. Keyword in a document is the most frequent word, and not in just one form but in all forms for its semantic meaning.

Keywords are terms that contain important information about document. Automated keywords extraction is a way to find a small set of words and phrases that describe the content of the document. Existing methods for automatic extraction of keywords can be divided into four groups:

- A simple statistical approach (statistical information about the words based on the frequency of words occurrence, tf * itf, mutual occurrence matrix, etc.).
- linguistic approach (lexical analysis, syntactic analysis)
- an approach based on machine learning (based on a set of documents with already extracted keywords and generating model to find key words in the new document)
- positional weighting approach (words at various positions have different degrees of significance) [5]

Frequently used algorithms for indexing *TfIdf*, requires a corpus of documents to obtain keywords. The other listed approaches for automatic word extraction require appropriate lexical resources or marked files.

In the absence of corpus, keywords can be extracted by number of occurrences of concrete terms in the given document. In order to single out the key words, the text must be normalized; in this case, normalization with n-gram has been used. As previously noted, the text is cleared of stop words and all no-letter marks, and divided into sentences. The most frequent terms are stored in a set of basic keywords. The document consists of sentences and the frequency of the term is calculated at the sentence level. Extracted keywords are the basis for further searching for keywords and phrases. If a term occurs more frequently in sentences together with certain key words, then it closely define that keywords and is also a candidate for a keyword.[7] Graph shows the distribution of the number of sentences in which some no-key terms appears in the same sentence as keywords appears. (Figure 4.)

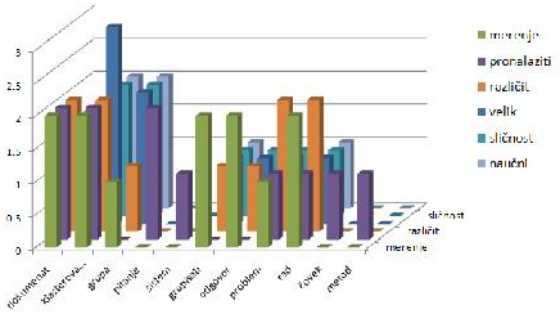

Figure 4. Mutual appearance key-terms and no-key-terms in a sentence

To extract these terms, co-occurrence matrix has been made. Matrix contains the number of sentences in which some of the key words(q) and other terms(w) occur together. If a term occurs selectively with only some keywords, that is an indication that term has greater significance. Calculating the deviation from the expected values was performed by h-square test. The expected value is calculated as the product of n_w, the total number of terms in sentences in which the observed term(w) appears, and p_q, as (the sum of the total number of terms in sentences where keyword g appears) divided by (the total number of terms the document) (Formula 1) [8].

$$\chi^2(w) = \sum_{g \in G} \frac{(freq(v_q) - n_w * p_q)^2}{n \cdot p_q} \qquad (1)$$

If there are groups of words that are often used together, their n-grams will be singled out. N-grams with the largest h-square test are additional keywords candidate. By allocating only the first n-grams we ignore the rest of the word, and it may be that some of the words that appear frequently could be overlooked due to different prefixes. On the other hand the meaning of different words can have the same prefix, so it will be included in the n-gram keywords. In order to these characteristics be detected, the n-gram analysis of words are made.(normalization 4). For the first set of key n-grams, only the initial n-grams, are taken, but the number of their appearance is calculated independently of their position in the text. Therefore, if the word longer than four letters is significant, it will be identified by extracting all the n-grams that appear in it, because they will show up a greater number of times in sentences in which the first n-gram of the word appears. In paper [9], the results of applying different normalization for extracting key words have been presented.

In this version of sistem the algorithm for extraction of keywords based on the normalization of 3 and 4 has been implemented. (Figure 5.)

```
procedure GetKeywords(DocumentId)
KeyWords[DocumentId]=0
Document3 = Normalization3(DocumentId)
foreach WordId∈ Document3
  begin
      WordNum[WordId] = CountRepeatWordInDocument(WordId, Document3)
  end
Avg = Average(WordNum)
Stddeev = StandardDeviation(WordNum )
foreach WordId∈ Document3
  begin
      If (WordNum[WordId]>Avg+Stddev)
            InsertKeyWords(WordId, DocumentId)
  end
Document4 = Normalization4(DocumentId)
foreach WordId∈ Document4
  begin
      WordNum[WordId] = CountRepeatWordInDocument(WordId, Document4)
  end
SentenceNum = NumberSentenceWithWord(WordId)
foreach WordId∈ (Document4 - KeyWords)
  begin
      HSquare[WordId]=0;
      foreach KeyId ∈ KeyWords[DocumentId]
        begin
            CoOccurrenceMatrix[WordId, KeyId]
            HSquare[WordId, KeyId]

        end
      HSquare[WordId]+= HSquare[WordId, KeyId]

  end
Words = GetTop40WordsInDocumentOrderByHSquare(Document4)
InsertKeyWords(Words, DocumentId)
```

Figure 5. Keyword extraction pseudo code

V. ANSWERING PROCESSOR-SIMILARITY SEARCHING

Described system need to return response on question inserted. Response can be pure response or whole document containing response. *Answering Processor's* main task is to find the most similar existing question to the question asked. If the most similar question to question ask doesn't exists, or user requests to get the whole document instead, it is necessary to find a document which has a response for asked question. Examining the similarity between questions or documents is done by cosine similarity algorithm. For calculating the cosine similarity value, we need data related to frequency number of a word in question in comparison with the frequency number of all word in question/document.

Pseudo code for calculating *TfIdf* is depicted on figure below (Figure 6)

```
procedure GETTFIDF(Word, Question, DocumentId)
wordNumber = CountRepeatingWordInQuestion(Word, Question)
words = CountWordsInQuestion(Question)
Tf=wordNumber/words

wordNumberInDoc = CountRepeatingWordInDoc(Word, Question)+ 1
wordsDoc = CountWordsInDoc(DocumentId)+1
Idf=log(wordNumberinDoc/wordsDoc)

TfIdf= Tf*Idf
return TfIdf
```

Figure 6. Pseudo code for calculating TfIdf

TfIdf is product of *Tf* and *Idf*. *Tf* is ratio of the number of concrete word occurrences in concrete question and the total number of words in that question. *Idf* is calculated with formula log((wnd+1)/(wd+1)); *wnd* is number of word occurrences in the document and *wd* is total number of words in document.

Pseudo code for calculating cosine similarity is depicted below. (Figure 7)

```
procedure COSINUSSIMILARITY(Question1, Question2, DocumentId)
SumTfIdfQ12=0
ProductTfIdfQ12=1
TfIdfQ11=0
TfIdfQ22=0

    foreach Word ∈ Question1
        begin
          TfIdfQ1=GetTfIdf(Word, Question1, DocumentId)
          TfIdfQ2=GetTfIdf(Word, Question2, DocumentId)

          TfIdfQ12 = TfIdfQ1* TfIdfQ2
          SumTfIdfQ12=SumTfIdfQ12+ TfIdfQ12

          TfIdfQ11= TfIdfQ11 + TfIdfQ1* TfIdfQ1
          TfIdfQ22= TfIdfQ22 + TfIdfQ2* TfIdfQ2

          SquaredTfIdfQ1=SQRT(TfIdfQ11)
          SquaredTfIdfQ2=SQRT(TfIdfQ22)

          ProductTfIdfQ12=ProductTfIdfQ12* SquaredTfIdfQ1* SquaredTfIdfQ2
       end

    if  ProductTfIdfQ12 != 0
          CosSimQ1Q2 = SumTfIdfQ12/ ProductTfIdfQ12
    else
          CosSimQ1Q2=-1

return CosSimQ1Q2
```

Figure 7. Pseudo code for cosine similarity

By using cosine similarity algorithm and any of previously described normalization, it is possible to define similarity between two questions. Following figure depicts cosine similarity for two questions. (Figure 8.). Cosine similarity between a question and a document is depicted below. (Figure 9)

```
procedure QUESTIONSSIMILARITY(Question1, Question2, DocumentId)
NormalizedQuestion1 = Normalize(Question1)
NormalizedQuestion2 = Normalize(Question2)
CosSimQ1Q2=CosinusSimilarity(Question1, Question2, DocumentId)
return CosSimQ1Q2
```

Figure 8. Cosine similarity for two questions

The procedure is following: first, questions asked have to be normalized. It is necessary to pass through a set of documents. For each document, you pass through the sentences, and calculate cosine similarity between each sentence and the question. If there are any similarity, cosine similarity will be greater than zero. In that case, sentence will be inserted into an array of similar sentences, and document will be inserted into an array of similar documents, on the same index position as sentence is written on its own array. At the end, sorting is performed on arrays in descending order. Due to the optimization, algorithm does not work directly with the sentences and documents, but with their ids. Data about number of words in the sentence and in the document are also stored in the database. This data is inserted into database just after document has been inserted. These data are necessary to calculate *TfIdf*.

```
procedure QUESTIONDOCUMENTSSIMILARITY(Question)
NormalizedQuestion = Normalize(Question)
MaxSimilarSentencesArray[0]=0
MaxSimilarDocumentsArray[0]=0
Counter=0;
   foreach Document ∈ Database
   begin
        foreach Sentence ∈ Document
        begin
          CosSim=GetCosinusSimilarity(Questions, Sentence, Document)
          if CosSim>0
             begin
               MaxSimilarSentencesArray[Counter]=Sentence
               MaxSimilarDocumentsArray[Counter]=Document
               Counter=Counter+1
             end
        end
   end
Sort(MaxSimilarSentencesArray)
Sort(MaxSimilarDocumentsArray)

return MaxSimilarSentencesArray, MaxSimilarDocumentsArray
```

Figure 9. Cosine similarity for question and document

VI. FUTURE WORK

The main purpose of previously described algorithms is to facilitate preparing process part of searching document algorithm. Existing normalization and algorithm for extraction keywords, reduces large lexical resource requirements. There are a lot of other peculiarities in Serbian grammar which aren't solved with n-gram normalization, e.g. synonyms, vocal changes, etc. Algorithm for normalization will be improved with these features.

System also should have additional part for clustering.

Keywords will be clustered by their frequency of occurrence and their distribution in sentences along with other terms. A chi-squared test and n-gram frequency of occurrence will be used for extracting n-grams for keywords candidates. After keywords are extracted, documents can be grouped by content. Also, we will optimize this algorithm for better performance in the future.

ACKNOWLEDGMENTS

This paper was partially funded by the Ministry of Education and Science of the Republic of Serbia for the project III-44007.

REFERENCES

[1] Fang, Z. (2002). E-Government in Digital Era: Concept, Practice and Development, *International Journal of The Computer, The Internet and Management*, 10(2), pp.1-22.

[2] Šimić, G., E. Kajan, Z. Jeremić, D. Randjelović. (2012). An Approach to Document Clustering using Hybrid Method, *IADIS e-Society Conference*, Berlin, March, 10-13, 2012, pp. 153-159.

[3] Živojin Stanojčić, Ljubomir Popović,GRAMATIKA SRPSKOG JEZIKA za gimnazije i srednje škole od 1. do 4. Razreda, Zavod za udžbenike

[4] Ejub Kajan, Aldina Pljasković, Adela Crnišanin „Normalization of text documents in serbian language for eficient searching in e-government systems“,ETRAN 2012, Zlatibor , jun 2012

[5] Munirul Mansur, Naushad UzZaman ,Mumit Khan “Analysis of n-gram based text categorization for Bangla in a newspaper corpus” , ICCIT 2006, Dhaka, Bangladesh, decembar 2006.

[6] D. Subotić, N. Forbes, “*Serbo-Croatian language – Grammar*”, Oxford : The Clarendon press, str.25-31, 61-64, 101-113

[7] Ulfeta Marovac, Ejub Kajan, Goran Šimić,” A solution of semantic clustering of text documents ”,CPPMI 2012, Novi Pazar, jun 2012

[8] Matsuo, Y. and M. Ishizika (2004). Keyword Extraction from a Single Document using Word Co-occurrence Statistical Information. *International Journal on Artificial Intelligence Tools.* Vol. 13, No 1, pp. 157-169.

[9] Ulfeta Marovac, Aldina Plja1sković, Adela Crnišanin, Ejub Kajan,” N-gram analiza tekstualnih dokumenata na srpskom jeziku”, Proceedings of TELFOR 2012, Belgrade, November 2012