Classification of Terms on a Positive-Negative Feelings Polarity Scale Based on Emoticons

UDC 811.163.41'322.2
DOI 10.18485/infotheca.2017.17.1.4

ABSTRACT: The goal of this paper is to draw attention to the possibility of using emoticon-riddled text on the web in language-neutral sentiment analysis. It introduces several innovations in the existing framework of research and tests their effectiveness. It also presents a software tool especially made for that purpose, explains how it builds a database with sentimental value of terms and offers the user manual. Finally, it presents a software tool that tests the new database and gives some examples of the analysis of the obtained results.
KEYWORDS: data mining, information extraction, emotions, text on the web.

PAPER SUBMITTED: 24 January 2017
PAPER ACCEPTED: 25 March 2017

Mihailo Škorić
miks@tesla.rcub.bg.ac.rs

University of Belgrade

1 Introduction

When creating natural language understanding software, there are two widely accepted approaches:

- Software that does not have a deep understanding of the meaning of written text, but only the grammar of the language that text is written on, which enables wider application.
- Software that has a deeper understanding of the meaning of the text, often limited to one or a small number of areas. This type of software is predominantly used for text classification.

Systems based on sentiment analysis assign sentiment values to text using multiple parameters, where a greater number of parameters means much greater complexity.

Under an assumption that reducing all possible parameters onto polarity, there isn't a significant information loss and that the machine can decide between multiple choices using a single parameter, task is reduced to determination of what is positive and what is negative.

With a goal of making a more cost-effective intelligent system, it would not be a good idea to ignore any available resources and that could potentially be useful. The main idea of this research is to use data mining methods for retrieval of metadata – in shape of determiners, that users of social networks inadvertently use in their messages (in the form of emoticons or language-universal phrases) and assigning values of sentiment polarity to terms in which those determiners are located. As the determiners are language-independent, the system would be language-independent as well. If it turns out to be valid, this method could allow machine learning the usage of huge corpus of texts that are pre-labeled with determiners.

1.1 Review of their former similar studies

In 2005 a series of experiments with the classification of mood for internet blog posts was published, and it served as the basis for many future studies (Mishne, 2005). These experiments used records of terms that appear in posts for which the authors themselves claimed were written in and reflected a certain mood. Term indexes were made and their frequency in posts that are equated with one of nine moods was calculated. Those nine moods were: amusement, fatigue, happiness, joy, boredom, a sense of succes, drowsiness, satisfaction and excitement. New posts were then tested on the frequency of indexed terms in order to determine the mood of the author who wrote them. Results were compared with human assessment of the same posts and it was concluded that the machine assesses the mood of the author only slightly worse than a human.

At University of Tokyo in 2009 a study was published, in which the nine moods were analysed in text using complex finite automata that recognizes the grammatical structure of the text (Neviarouskaya et. al., 2009). That same year, researchers at Stanford University in California unveiled the new web posts analysis system that uses algorithms that are trained to recognize emoticons and classify moods of *Twitter* posts as either positive or negative (Go and Bhayani, 2009). The goal was to create a system for the classification of posts, so that consumers can explore the attitude of previous customers before buying a product. Several different algorithms for machine learning

were trained with eight emoticons (five positive and three negative), and the results showed accuracy above 80% for guessing the mood of the posts.

Researchers at the Hebrew University in Jerusalem in 2010 carried out another similar research of sentiment expressed in *Twitter* posts, taking into account 15 emoticons, and 50 tags,[1] as addition, which is their original cotribution (Davidov et. al., 2010). The algorithms that were trained using tags, also succeeded in recognizing the mood of the post.

In the above mentioned studies emoticons in text are treated as character strings of explicit meaning. A different approach was proposed in 2010 by researchers at Hokkaido University in Sapporo. Dissatisfied with existting databases of emoticons and their values, they primarily dealt with how to determine their values more precisely. The idea was to treat emoticons as structures composed of separate elements that represent the eyes and the mouth. Composite parts were processed serarately to calculate their value. When the database was being made emoticons were classified by ten possible feelings: anger, resentment, excitement, fear, affection, happiness, relief, shame, sadness and surprise (Ptaszynski et. al., 2010). This study was later expanded, and in 2011 a paper was published on the research in which emoticons were defined as parts of natural language, so it was suggested that their research is should be included in natural language research (Ptaszynski et. al., 2011).

1.2 Basic information about the experiment

Goal of the experiment is to test a new approach to text extraction using emoticon extraction in a new way, by combination of the following three ideas:

- Emoticons that will be used in the experiment will not be assigned discrete values such as positive or negative, but values on a scale from most positive to the most negative. This will be also reflect on the terms whose values will also be assigned on this scale.
- Only universal and language-neutral determiner strings will be used. Goal is to create a fully language-independent system that would greatly broaden the possible corpus.

[1] Users of *Twitter* platform have an option to additionally mark their posts with tags so that posts that talk about a certain topic can be found more easily using tag serach.

– Instead of separate messages, such as those on *Twitter* platform, messages that are a part of a conversation will be used. The goal of this is to test the ability of the determiner to influence not only the message in which it is located, but also the messages in the immediate vicinity.

An additional outcome of this experiment are two software tools, specifically designed to perform the experiment. These applications should enable future researchers to test similar ideas, and a detailed explanation of the software could help in the development of entirely new, better and more complete tool.

The main idea of this experiment is to prove that it is possible to:

– build an inverted index of terms in a language-neutral way using a corpus of texts that contain known determiners.
– automatically assign values to terms on positive-negative scale using those determiners, so that specific values reflect the attitude of the people using specific terms.

The experiment consists of two parts:

– creation of a database containing an inverted index of terms and their values, using software tool specifically made for this experiment.
– testing values of the terms from the database by comparing them with human assessments and by using the software testing tool that is specifically designed for databases that are the product of the first part of the experiment.

2 Preparation for the database creation

Database which is a product of this experiment brings terms (words that appear in conversations) and values they reflect their polarity (if any) in numerical form together in one place. The values are calculated by taking into account the proximity of the determiners to the observed term and the value of those determiners. The determiners can be either emoticons or phrases that appear in the conversation, which by nature are not of universal meaning and reflect a positive or negative attitude, replacing facial expressions and/or intonation in the written text.

The intensity values of a determiner directly affects the intensity value which he transfer onto the term. Also, the closer the determiner is to the term, the more its value affects the value of the term.

For the database formation to be successful and its final outcome satisfactory, three prerequisites should be met:

- collected corpus must be organized in a certain way;
- collected texts and messages must contain determiners that would help assign a value to a nearby term;
- determiners must have a predetermined value.

In the following chapters it will be explained in detail how the databases for this study were created, from the collection of the corpus to the export of completed database, which can then be used in several ways.

2.1 Collecting textual corpus

The basic idea was for the database to be based on a corpus of texts containing determiners which express positive or negative attitudes of interlocutors. Messages used for this experiment were gathered from chat histories files of *Facebook* users, for which a suitable preprocessing XLS transformation was prepared (explained in chapeter 3.1). Several volunteers downloaded these files from the official web page[2] and forwarded them to the author of this work in order for them to be included in the corpus used for the research.

Six ZIP files (from six users) sizes from 1.85MB to 167.09MB were collected. Together, those files contain 3,884 conversations of 2,019 different users and 1,843,826 messages that those users exchanged. The content of these messages is used in making the database for this research. With this, the first two aforementioned prerequisites were fulfilled.

2.2 Assigning values to the determiners

The values of determiners were obtained by a method of survey which was conducted on the internet platform *Google forms*, which people were able to access through hyperlinks published on social networks or passed from the other participants. Respondents were tasked with asigning values between 0 and 10 to a set of chosen emoticons and phrases, where 0 represented the highest intensity of negative mood, and 10 the highest intensity of positive mood. They were told to consider before rating how a determiner reflects

[2] Social network *Facebook* allows all its users to download a single ZIP file that contains all of their current multimedia files and chat history via settings on personal profile tab.

their feelings or mood when they send it or how they perceive it when they receive it in a message. Respondents also had the opportunity to propose additional determiners that they consider to be relevant and suggest its value.

During a period of 30 days, 389 survey participants assessed the 19 submitted and proposed additional 22 determiners, 9 of which were accepted[3]. The results differed from one determiner to another – somewhere the results were more balanced, while somewhere they were more speckled (Figure 1). After the survey ended, the value of each determiner was calculated as the arithmetic mean of a set of values that the respondents submitted (or proposed for 9 subsequently added determiners).

In order for values to better reflect what they represent – a point on a positive negative scale – they were mapped from set $(0, 10)$ onto set $(-1, 1)$, which was done by applying the formula:

$$x = (x - 5)/5 \tag{1}$$

Thus, the value of which corresponds to a negative mood of highest intensity became -1, while the value corresponding to the highest intensity of the positive mood became 1. Also a universal correction factor that multiplies all values was calculated. This factor was equal to the ration of the highest value 1 and tha value of a determiner with highest positive value 0.81, so that determiner *<3*, which, according to participants of the survey reflects a positive mood of the highest intensity, was assigned a maximum value of 1. Using this all determiners were assigned proper values and the last prerequisite for database forming was fulfilled.

3 Database construction

Software, developed specifically for this research was used to create the database. It was written in C# programming language and it can be ran on *Windows* platform, using any version of the operating system that runs to 64 bit. The interface is user-friendly and entirely in Serbian. The goal in designing this software was that any researcher who speaks Serbian can use it independently, create a new database or use it for a new research.

[3] A total of 143 users actually suggested a determiner, and the necessary number of proposers of a new determiner was 48, or more than one-third. If enough users suggested a determiner its value was calculated as the mean of all the proposed values and it was involved in the study.

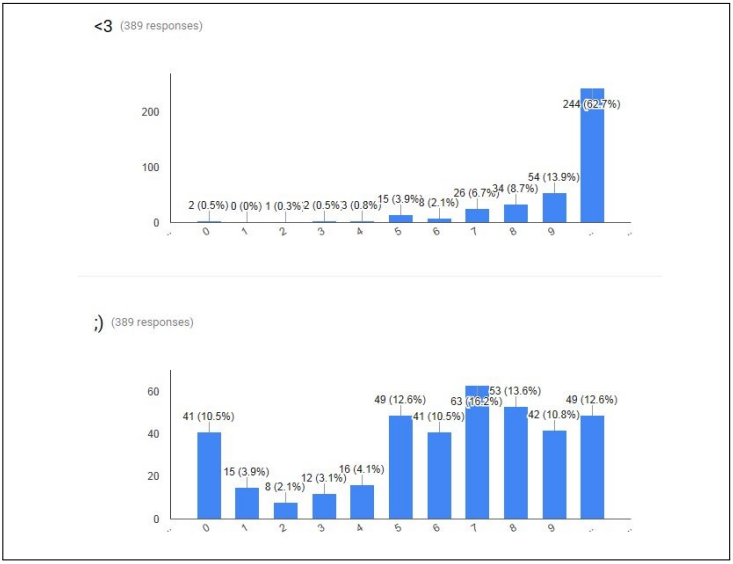

Figure 1. An example of the different level of agreement of users considering the value of determiner *<3* (higher agreement – over 60% of respondents agree on one answer) and determiner *;)* (lower agreement – five most commonly chosen responses in range of just 5.7%).

Software's task is to read the input file (in proper form), and trough seven steps, transform it into a database that contains terms found and their value on a positive-negative scale depending on the determiner that are in the vicinity of the respective term. In the following chapters the all steps of the database creation will be explained.

3.1 Preprocessing

Preprocessing is performed on the file that the user selects by pressing a button *Pronađi datoteku* (Open file). That opens a *Windows Explorer* catalogue in which user selects a file from the local computer in the usual way. The tool currently supports only files that contain chat history of *Facebook* social network, and the only file format available for selection is *htm*. The

determiner	meaning	value	determiner	meaning	value
:))	smile	0.56	:)	smile	0.26
:D	grin	0.91	:P	tongue out	0.35
:p	tongue out	0.24	xD	grin	0.59
:o	wonder	-0.22	:O	wonder	-0.29
:((	sad face	-0.86	:(	sad face	-0.66
:/	peeve	-0.48	:\	peeve	-0.48
:**	kisses	0.88	:*	kiss	0.80
hahaha	laugh	0.72	haha	laugh	0.15
<3	heart	1.00	;)	wink	0.26
:'(	cry	-0.74	lol	laugh	0.04
^^	joy	0.95	-.-	speechless	-0.45
:3	cat	0.94	*.*	glint	0.95
:S	peeve	-0.05	:'D	fall about	0.95
o.o	disbelief	-0.10	...	speechless	-0.18

Table 1. The final list of determiners and their assigned values.

user continues by pressing the button *Preprocesiraj i transformiši korak po korak*[4] (Figure 2).

Because the software uses XSL transformations, the basic precondition is for the input to be a well formed XML file. User picked file (Figure 3) is first purged of all the characters that may prevent it from being well-formed. It's done using the *Regex.Replace* function and regular expression *[\u0000-\u001F]*, which finds the first 32 characters from *ASCII* set and then replaces them with an empty string. To further ensure that the document is well-formed character *&* is replaced with a whitespace and a new root element is introduced.

Preprocessing is finished with an XSL transformation that transform a document into one that can be additionally processed (Figure 4). All nodes whose children do not contain user-exchanged messages are removed. Only node kept is * * (Figure 3), which contains the messages. These nodes are constructed, a new temporary file with a current document is saved and the preprocessing ends.

[4] Alternativly, he can use *Izvrši kopletnu transformaciju prema podrazumevanim podešavanjima* (Execute complete transformation using default settings) button, and start a complete transformation using default settings, or the last settings used in the step by step transformation.

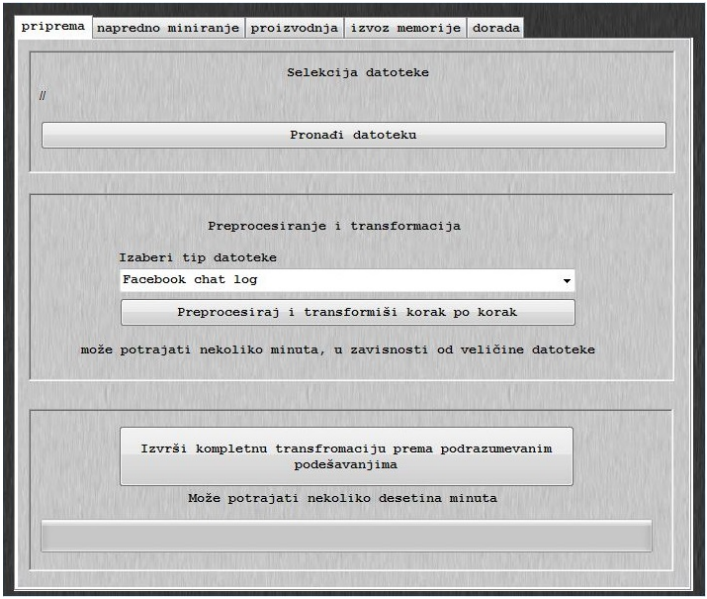

Figure 2. The appearance of user interface software for text mining.

3.2 The extraction of values from the document

Determiners extraction is performed in two steps. The first part of the extraction consists of picking determiners that will be searched for and the values that they stand for. Determiners and values can be set one by one, or this step can be skipped. In case this step is skipped, the software will automatically load the default determiners and their values (Table 1).

All default determiners and their values listed in the advanced text mining catalog (Figure 5). The values can be changed manually via the text field next to each term. If any determiner is undesirable, its value should be replaced with "/" character and it will not be used in text mining. In case the user wants to test a new determiner, he can use *dodaj emotikon* (add emoticon) button at the bottom of the page, after he fills the necessary fields – term string ant it's value. If the entered value of a determiner is not in the range of -1 to 1, entry will not be accepted and the user will receive a message about a failed new determiner addition. The maximum number of determiners that can be used is 36.

```
▼<html>
  ▼<head>
     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
     <title>Korisnik 1 - Messages</title>
     <link rel="stylesheet" href="../html/style.css" type="text/css"/>
   </head>
  ▼<body>
    ▼<div class="nav">
       <img src="../photos/profile.jpg"/>
     ▶ <ul>...</ul>
     </div>
    ▼<div class="contents">
       <h1>Korisnik 1</h1>
      ▼<div>
       ▼<div class="thread">
          Korisnik 1, Korisnik 2
        ▼<div class="message">
         ▼<div class="message_header">
            <span class="user">Korisnik 1</span>
            <span class="meta">Tuesday, February 9, 2016 at 2:44pm UTC+01</span>
          </div>
         </div>
         <p>Tekst poruke</p>
        ▼<div class="message">
         ▼<div class="message_header">
            <span class="user">Korisnik 2</span>
            <span class="meta">Tuesday, February 9, 2016 at 2:39pm UTC+01</span>
          </div>
         </div>
         <p>Tekst poruke</p>
```

Figure 3. A fragment of the input file.

```
▼<root>
  ▼<conversation>
    ▼<messages>
      ▼<message>
          <sender>Korisnik 1</sender>
          <text>Tekst poruke</text>
       </message>
      ▼<message>
          <sender>Korisnik 2</sender>
          <text>Tekst poruke</text>
       </message>
```

Figure 4. Fragment of the input file after preprocessing.

Advanced users can modify text file *data/emoticons.txt* which contains all the default determiners and their values. At any time, determines can

Figure 5. Settings example for extracting value from the text.

be restored to default settings of the above file, or new settings can dubbed over existing ones. Both options are activated by pressing the appropriate buttons located on the bottom of the screen (Figure 5). In both cases, due to possible irreversible data loss, the user will need to confirm the process in an additional dialog box that will appear on the screen.

Once the user is satisfied with the settings he can switch to the second part of the extraction, which is done by pressing *ekstraktuj vrednosti iz teksta* (extract values from the text) button on the production tab (Figure 7). Before pressing this button the user can also, optionally, check the field *Koristi predodređene regularne izraze za poboljšanu pretragu* (Use predetermined regular expressions for improved search), which should lead to a greater number of determiner extractions in the text.

Execution begins with optional fields check. If the field is checked, document goes through several *Regex.Replace* functions that search for known deviations of several determiners. but first, automatic mapping of characters in XML are reversed, as this part of the transformation doesn't require

well-formed XML document. Entity reference *lt;*(character *&* that marks the begining of the entity reference was replaced with a whitespace during preprocessing step) is replaced with *<* character, so that the emoticon *<3* cound be found in the text. Entity reference *039* is replaced with *'* character, so that emoticons such as *:'(* could be found. String *:-* is replaced with character *:* , so that both characters with and without *nose* such as *:-D* or *:-(* could be mined. All the Cyrillic characters used in default determiners are transliterated into Latin in order to find the emoticons in the messages of users who use the Cyrillic alphabet (*:Д*, *xД*). All variants of emoticons *xD*, *:S* i *o.o* written in either upper or lowercase are converted into its basic format. Hyperlink beginnings *http://* and *https://* are replaced with a neutral character string *yy*, so that they wiil not be mistaken for *:/* emoticon. Finally, regular expressions *[a/h/A/H][a/h/A/H][h/H][a/A][h/H][a/h/A/H]* and *[h/H][a/A][h/H][a/A]* are used to find as many different examples of expression of laughter, and the expressions found are replaced with *hahaha* and *haha* respectively.

If this option is not checked, this step will be skipped and only determiners in its default form, which is listed on the advanced mining tab, will be found and processed.

The mere extraction involves loading selected terms and their values from advanced mining tab in two arrays, and then then processing them. Each of the elements of the first array, expressions array, pass through a *for each* loop in which each of its appearances in the text is replaced with *<emot value='x'/>* node, representing an empty XML node in which x is the value of the current determiner which is loaded from the second array, values array. If the value found is */* instruction will not be executed, and if the value of an expression is not between -1 and 1 the program will report an error and stop the execution.

After each of the determiners is replaced with a node whose attribute is its value (Figure 6), XML document becomes well-formed again and is ready for further processing. The user will receive a message that the extraction is completed, and he can move onto the next step.

3.3 Assigning values to segments of text

In the experiment segment of text or a sentence is equated with a sole message sent from one user to another. The basic idea is that the values of determiners found in the message reflect the value of the message, and the surrounding messages in some special cases. Depending on the message

```
▼<message>
   <sender>Korisnik 1</sender>
   <text>tekst :)</text>
 </message>

▼<message>
   <sender>Korisnik 1</sender>
   <text>teskt <emot value="0.26"></text>
 </message>
```

Figure 6. Message appearance before and after extraction of determiners.

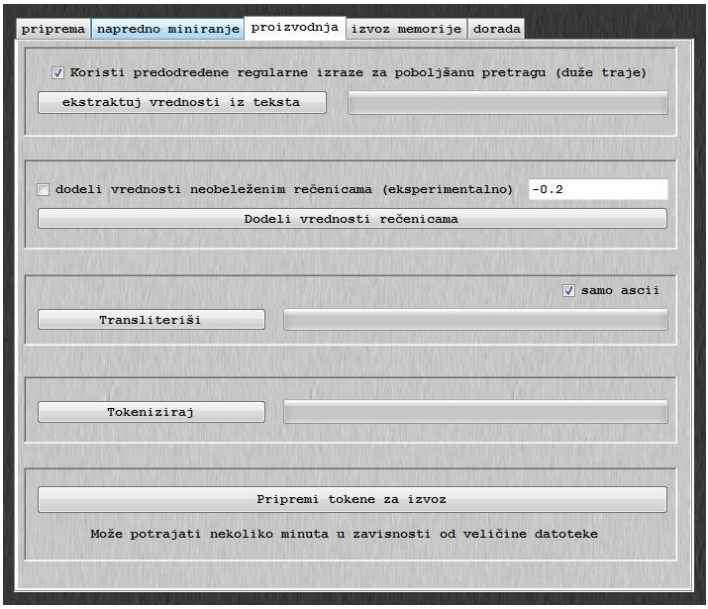

Figure 7. Step by step database production tab.

text content (beside the determiner[5]) and the content of the previous and following messages, there are three types of segments values allocation:

[5] In this case, all messages that contain less than 4 characters are treated as empty.

1. if the message contains text beside the determiner and the following message also contains text – only determiners found in the message affect its value Examples:
   *A: Happy birthday :**
   B: Thank you very much
   Determiner *:** will affect the segment value, while the message of person B has no effect on it, because it does not contain a determiner. *A: Happy birthday :**
   B: Thanks :)
   Determiner *:** will affect the segment value, while the message of person B has no effect on it, because the determiner in that message refers to the text in the same message

2. if the message contains text, but not a determiner, and the following message contains determiner but not text – determiner will refer to the previous message. Example:
   A: I missed the bus
   B: :(
   Determiner *:(* from the message of person B will refer to the message of person A, becouse the second message does not contain text, and its determiner must refer to previous message.

3. if the message contains both the determiner and the text, and the following message contains determiner but not text – determiners from both messages will refer to the message that contains text. Example:
   A: I missed the bus -.-
   B: :(
   Both determiners *-.-* and *:(* will refer to the message of person A, becouse the second message does not contain text, and its determiner must refer to previous message.

There is also a forth possibility – assign values to messages no determiners affect. This feature is based on the assumption that the absence of determiners also affects the sentiment (by intuition, negative one). The disadvantage of this option is that we can not be sure that the absence of determiners really bears (negative) meaning, and the assessment of value is also difficult task. An additional problem is that the message may contain a determiner that was not used in the text mining tab. Therefore, this possibility is optional, and the value it assigns is not predefined. If the user wants to assign

a certain value to emph unlabeled messages, he can do it by checking *dodeli vrednost neobeleženim rečenicama* button and by filling respective text box with value between −1 and 1 (figure 7).

```
▼<root>
  ▼<conversation>
    ▼<messages>
      ▼<message>
          <sender>Korisnik 1</sender>
          <text>tekst poruke 1 <emot value="0.26"><emot value="-0.26"></text>
       </message>
      ▼<message>
          <sender>Korisnik 2</sender>
          <text><emot value="0.15"></text>
       </message>
```

```
▼<root>
   <emotext value="0.15">tekst poruke 1</emotext>
```

Figure 8. The appearance of the document segment before and after the value was assigned – both messages contain determiners, but the second one does not contain text (case 3)

The mere assignment of values is done via XSLT transformation. First, for each *labeled* message the arithmetic mean value of all the determiners that refer to it is calculated, and this value is written into its new attribute node *value*, while the earlier made determiner nodes are removed. Then, if the optional box was checked, previously assigned messages are assigned value from the optional text box. The result of the transformation is an XML document containing root element and in it nodes of all messages that have an assigned value. Upon completion of the transformation the user will get the message that the assignment is completed and green light to move on.

3.4 Transliteration and reduction

These two steps are done together by pressing *transliteriši* (transliterate) button of the production tab. The purpose of this step is to get a clean out text for the creation of the database. In this case, the text is considered to be

any string that contains only alphanumerics. Optional checkbox *samo ascii*, serves to further limit the collection only to alphanumeric characters from the ASCII character set.

The reason this is optional is that it introduces nearly as many problems as it resolves. Assuming that not all social networks users use the full capacity of Unicode character set, but also *cropped Latin*[6], a problem occures where strings *istraživanje* (*research*) and *istazivanje* (*research*) will not be recognized as same, but different words. Converting all characters into ASCII set resolves this problem but introduces a new problem where words that would differ in Unicode set would be recognized as same – for example *španac* (*spainiard*) and *spanać* (*spinach*), would be recogniyed as same. This option is therefore best to be used depending on the situation.

Before proceeding further processing, the current XML document goes through XSL transformation in which all uppercase letters are changed to lowercase using *translate* function, so that words would be recognized no matter the capitalization of letters.

The second step of the text clean-up, is cleaning all the characters that are not letters or numbers, eliminating the possibility that the word contains any punctuation or other non-alphanumeric character. Again by using the *translate* XSL function all unwanted characters (all characters except for the small Latin letters and digits) are converted into whitespace. This process can result in incorrect words if the user accidentally entered an undesirable character during typing (*Mar?ia*) or if he deliberately used a special character (*M@ria*). In the first case, the string will become emph Mar ia, and in the second *M ria*.

3.5 Tokenization

User starts this step by pressing the *tokenizacija* (tokenize) button of the production tab (Figure 7). Tokenization is done by using whitespace as a delimiter, so a token is defined as any character string between the beginning of the message and the first whitespace, any character string between two adjacent whitespaces, or any charachter string between the last whitespace and the end of the message. It is carried out on an XML document level, with each token getting its own XML node. The first part of tokenization is done using *Regex.Replace* function in three steps:

[6] Latin letters without diacritics, such as: *c*, *z*, *s* instead of *č*, *ž*, *š*.

- First, the beginning of each message in an XML document is found. It is done by searching for the string *">*, which occurs only at the end of an opening tag of elements that contain an attribute, in this case each element that contains a single message. As the root element does not contain an attribute, it will not be found. After each of the found expressions string *<token>*, which marks the beginning of the token, is added.
- Then the end of each message is found, by searching the string *</emotext>* – which is the ending tag of each message node. After each of the found expressions string *</token>* is added, to get string *</token></emotext>*. With this step, each message in the XML document becomes one token: *<emotext value="x"><token> ... </token></emotext>* .
- The last step is to divide the interior of each message into tokens, using whitspaces. To avoid unwanted replacement within the opening tag *<emotext value="x">*, for whitespace between the name of the element and the name attribute, each *emotext value* string is replaced with *emotext_value*, and then, each whitespace in the document is replaced with *</token><token>* string. Character _ becomes whitesapce again and document structure becomes: *<emotext value="x"><token> ... </token><token> ... </token></emotext>* .

The second part of tokenization is performed using four additional XLS transformations that filter selected tokens and give them value:

- The first transformation removes all messages containing more than forty tokens, in order to avoid awarding the value to all words in, for example, a text pasted into chat, which is not an active messages in the conversation.
- Second transformation deletes all tokens containing less than two characters (assuming they do not carry any meaning, because they either represent a non-functional word or were created by breaking large character strings such as hyperlinks), and all the tokens that contain more than twenty characters (assuming that the vast majority of them is random).

After the second transformation, another operation is needed, introducing of an additional token without value to the last place in the document. This is done by inserting *<token>ERR0001</token>* string before the closing tag or root node in order to get *<token>ERR0001</token>*. The meaning of this operation will be explained in the following section.

– The third transformation is used to make tokens independent. Initially every token is assigned witha a value of a parent message which is allocated in a new attribute for each token, and then all messages tags are removed, leaving only the root node and in it token nodes with associated values. It also sortitra all tokens in alphabetical order in relation to their text content, and assignes each token with a new attribute *no*, whose value becomes the ordinal number of the token in the document, defined by *position* XSL function.

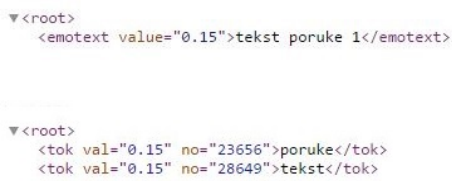

Figure 9. The appearance of a document fragment before and after tokenization.

– Fourth transformation does not contribute to the structure of the document, but only the speed of its processing. Name of each element *token* is shortened to *t*, and each attribute *value* to *v*, which greatly reduces the size of the file and accelerates the further processing of the document.

3.6 Creating index of tokens with values

This step is performed by pressing the *pripremi tokene za izvoz* (prepere tokens for export) button in the bottom area of prduction tab (Figure 7). Execution of this command is possible only after all the preceding steps of preparation were completed.

For index to serve its purpose, all terms in it receive two attributes: the mean of all values that refer to this token in the used corpus and the number its repetitions. Creation of the index is done by using six steps XLS transformation of an XML document that contains tokens:

– In the alphabetically arranged documet the first node, x, is found, and the first following node with different content, y, is found.
– Their ordinals are fetched from the *no* attributes. For x values is assigned to variable n, and for y to varable m.

– Text of the x node is inserted into a new XML document for inverted index, in element t (token).
– New element t is assigned with attribute c (count), which indicates the number of repetitions of this token in the database, and its value is equal to the difference of the ordinal numbers from the two nodes.

$$c = m - n \tag{2}$$

– New element t is assigned with attribute v (value), indicating the value of the token on a positive-negative scale, which is equal to the arithmetic mean value of attributes emphv for all tokens containing the same exact text.

$$v_x = \frac{\sum_{i=n}^{n+c-1} v_i}{c_x} \tag{3}$$

– Node y becomes the first node in the document and the procedure starts again.

This procedure assigns attributes to all the tokens except for the last one, *ERR0001* token, which was created during tokenization of the document and is used for marking the end of the document, and for calculating the attribute c for the previous sibling token.

t (term)	v (value)	no (ordinal number)
zdravko	-0.1	859231
zdravko	0.3	859232
zdravlje	0.26	859233

t (term)	v (ordinal number)	c (repetitions number)
zdravko	0.2	2
zdravlje	0.26	1

Table 2. Appearance of database fragment before and after the final transformation.

Having successfully carried out the conversion of a set of tokens with the values to the final inverted index of terms with its values (Table 2), the user will get the message about the successfull completion, and the index will be stored in a temporary file until his final export.

4 Database update and export

Pressing the *Izvezi u novu datoteku* (Export to new file) button of the export tab (Figure 10) opens *Windows Explorer* file saving catalogue. The format in which the file will be saved is XML and the user chooses the name and location. Content is copied from the temporary file created in the previous step, and it represents the final database.

In the event that the user has already processed some files, he also has *Dopuni postojeću bazu* (Update existing base) available (Figure 10). It allows the data to be drawn from the new corpus and update previously exported databases. In this case, pressing the appropriate button opens the *Windows Explorer* open file catalogue. The user needs to mark the file he wants to update, and updated database is created in four steps:

Figure 10. Appearance of export/update databases tab.

– The selected file is loaded and added to content of previously created index stored in a temporary file.

- String *</root><root>* is found and removed. This results in a document with one, instead of two root elements.
- XSL transformation is used re-sort all tokens in alphabetical order.
- As in this case there can't be more than two tokens with the same text, their unification is simpler. If token does not have a pair, it is copied, and if it does it gets new attribute values. Value attribute is calculated as the quotient of the sums of products of attributes v and c of both elements and the total number of their repetitions (2.4). Repetitions attribute is equals the total number of their repetitions (2.5).

$$v = \frac{v_1 * c_1 + v_2 * c_2}{c_1 + c_2} \tag{4}$$

$$c = c_1 + c_2 \tag{5}$$

  The new elements with the attributes are kept while the old ones deleted from the index, and thus only one copy of each token with a specific attribute remains.

The new document replaces the one that was marked in the open file catalog, so this is an option that should be used carefully. Upon completion, the user will receive a message about successfull completion.

If the user wants to manually review the newly created database, and does not feel comfortable in an XML environment, he can export it into CSV (comma separated values) format that can be viewed using *Microsoft Excel*, *Open Office Calc* or similar software for working with tables. He does this by pressing the last, fourth button on the export/update tab.

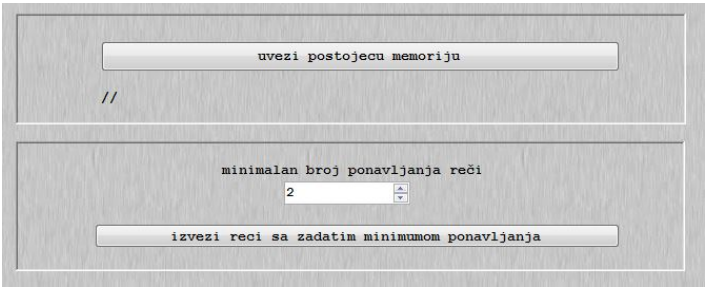

Figure 11. Appearance of database finishing tab.

The database can be plucked based on repetition number of the terms inside of it. This option is located on the fifth software tab, experimental database finishing tab. The user must first load the database by pressing *uvezi postojeću memoriju* (import existing database) button, and then chooses how many times the term bar needs to be repeated to be taken into account (Figure 11).

If the user, for example, choses number 2, by pressing the *izvezi reči sa zadatim minimumom ponavljanja* (export terms with specified minimum repetition) button, a new database will be formed without all the terms repeated less than 2 times (once), and will be saved in the desired place. User should use this option if he considers it necessary that the term appears a number of times before it is considered representative.

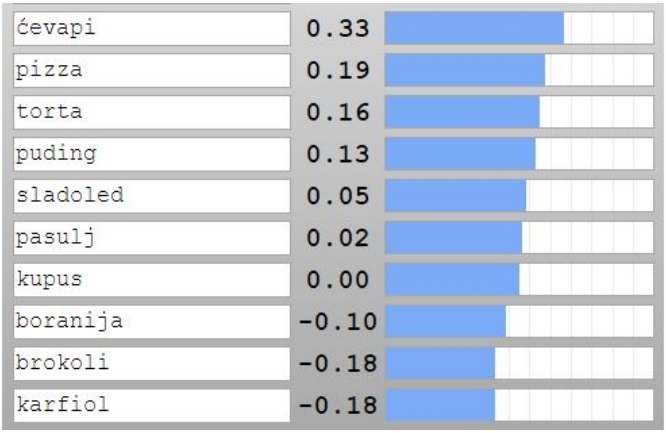

Figure 12. Example of value comparison of terms from the database using a web application – fast food, desserts, vegetables.

Once the user has finished creating the desired database and successfully exported them, they can be tested manually with the help of search option built into any software for working with text or XML documents. Also, the base in any of its forms can tested using software specially-made for this experiment (Figures 12 and 13).

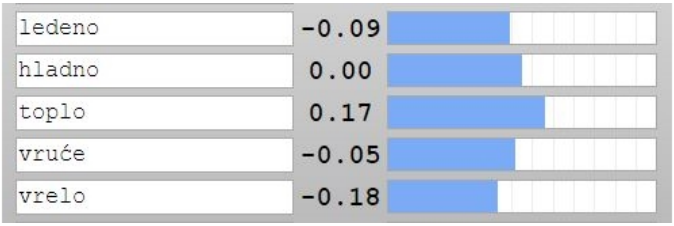

Figure 13. Example of value comparison of terms from the database using a web application – temperatures.

5 Conclusion

5.1 Review of the experiment outcome

Despite relatively small sample of 1,843,826 messages, from roughly estimated 500 billion messages belonging to the users of *Facebook* social network in Serbia, the experiment met its objectives. Based on the extracted only twenty-eight determiner strings and with the help of seven regular expressions that find their variations, and without prior knowledge of the language or its grammar, software formed a database of terms and their values on a positive-negative scale.

The system has been tested by several independent evaluators and based on their responses it can be concluded that the values are non-random and representative, hich implies that this method of terms classification is possible for any natural language using determiners, but it is still necessary to carry out a systematic evaluation.

The algorithms used are still far from perfect, and in there is plenty of room for improvement and progress considering them. Documented process of the extraction and transformation should contribute to future studies of the same or similar ideas. The next step can be extracting an expanded set of determiners, operation over an extended set of texts or, ideally, an expanded set of parameters that should be evaluated. Corpus expanded, for example, to other natural languages would greatly contribute to the volume of terms in the index, while the expanded set of parameters could add a new depth of understanding of the text by both the machines and the people.

5.2 Possible applications

This method of determiner extraction and generally similar researches can find a wide variety of applications divided into two groups.

Social and demographic research:

- Marketing research: exploration of current vs alternative approach to the marketing of products and services. This is the most common use of similar studies primarily for financial reasons, because using these companies can save money or get a new influx of goods.
- The public opinion research: what people like, what they do not like, what are their opinions about things or ideas that text refers to. It can be used in different ways in social and demographic research to quickly and efficiently collect large amounts of information.

Developing intelligent systems that work with information:

- Information retrieval: retrieval of specific information in the text, as well as finding information that can not be precisely defined. Classification of texts according to the mood that is expressed in them to help find the necessary information.
- Natural language understanding and analysis: understanding of written text and text queries, analysis of moods in the text, processing of digital linguistic resources such as automatic parallelization and automation of any operation that requires a deep understanding of the written text.
- Artificial intelligence: automated conversation in natural language and work with clients.

5.3 Possible deficiencies and improvements

- The problem of possible random responses during the survey:
  Create a system that rejects the member who answered at random, for example, using an introduction of additional questions that requiring specific responses. In that way careless users who do not read the questions are more likely to be identified.
- Systematic evaluation of the results:
  Conduct a detailed systematic evaluation in order to determine the credibility of the results.

References

Davidov, Dmitry, Oren Tsur, and Ari Rappoport. Enhanced sentiment learning using twitter hashtags and smileys. In *COLING '10 Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, 241–249. Association for Computational Linguistics, 2010.

Go, Alec, Lei Bhayani, and Richa nad Huang. Twitter sentiment classification using distant supervision. Processing, 2009.

Mishne, Gilad. Experiments with mood classification in blog posts. In *Proceedings of ACM SIGIR 2005 workshop on stylistic analysis of text for information access*, Vol. 19, 321–327. Citeseer, 2005.

Neviarouskaya, Alena, Helmut Prendinger, and Mitsuru Ishizuka. Compositionality principle in recognition of fine-grained emotions from text. In *Proceedings of the Third International ICWSM Conferenc*, 278–281. The AAAI Press, 2009.

Ptaszynski, Michael, Rafal Rzepka, Kenji Araki, and Yoshio Momouchi. Research on emoticons: Review of the field and proposal of research framework. In *The Seventeenth Annual Meeting of The Association for Natural Language Processing*, 1159–1162. The Association for Natural Language Processing, 2011.

Ptaszynski, Michael, Pawel Dybala, Radosalw Komuda, Rafal Rzepka, and Kenji Araki. Development of Emoticon Database for Affect Analysis in Japanese. In *Proceedings of the 4th International Symposium on Global COE Program of the knowledge Federation*, 203–204, 2010.