# A text mining system for creating electronic glossaries with application to research of Church Slavonic language

N. Yu. Shokina, S. Mocken

Computing Center, Albert-Ludwigs-University Freiburg, Freiburg im Breisgau, Germany

Corresponding e-mail: nina.shokina@gmail.com, susanne.mocken@rz.uni-freiburg.de

In this paper we present a text mining system for creating electronic glossaries with application to research of Church Slavonic language. Preprocessing, core text mining operation (pattern discovery), and postprocessing (automatic and manual lemmatization) are described in detail. An original pattern recognition algorithm for the initial discovery of text elements and the subsequent identification of the positions of tagged elements (or sets of elements) within a TEI XML entry is presented. The application of software design principles to the creation of our software is briefly described.

*Keywords*: digital humanities, text mining, pattern recognition, lemmatization, software engineering, software design, TEI XML, Church Slavonic.

## Introduction

Digital Humanities is the intersection between computing and humanities, currently being a rapidly developing research area. The history of Digital Humanities started in 1949 [1], but until now there is no standard definition for it. There even exists a web-site which provides the definitions from participants of the so-called Day of Digital Humanities in 2009–2014 [2]. Every refresh of the page brings a new definition, and the creator of the web-site states that the database contained 817 variants in January 2015. One frequently used definition of Digital Humanities is given by an American scholar Kathleen Fitzpatrick: "For me it has to do with the work that gets done at the crossroads of digital media and traditional humanistic study. And that happens in two different ways. On the one hand, it's bringing the tools and techniques of digital media to bear on traditional humanistic questions. But it's also bringing humanistic modes of inquiry to bear on digital media. It's a sort of moving back and forth across those lines, thinking about what computing is, how it functions in our culture, and then using those computing technologies to think about the more traditional aspects of culture" [3].

Digital Humanities includes various interdisciplinary tasks, for instance, at the intersection of computer science and linguistics. The research of human cultural heritage entails collecting a variety of materials (charters, manuscripts, books, etc.) and keeping them in an appropriate place. Until a few years ago, archives and libraries were the main institutions that provided access to older and contemporary documents. Doing research on site took and still usually takes a lot of time and effort, especially if resources are either not available in

one place or the research questions cover a huge field. These conditions have significantly improved since more and more documents have been digitized and became freely accessible via the World Wide Web. Providing searchable online databases not only contributes to preserving cultural heritage and making it available to a wider audiences it facilitates scientific research as well.

One of the tasks set for the BMBF (Bundesministeriums für Bildung und Forschung) interdisciplinary project SlaVaComp (COMPutergestützte Untersuchung von VAriabilität im KirchenSLAvischen) [4] was to enable research of the geographical and chronological differentiation of Church Slavonic language between the 10th and 16th centuries. The underlying data consists of fifteen Church Slavonic-Greek glossaries that cover various regions and times. Transferring the data into an electronic database gives fast and readily available answers to a great number of linguistic questions. The current work is one of the first attempts to transform printed dictionaries into the digital XML format for further use in the Slavonic studies.

The electronic glossaries are marked up using the Extensible Markup Language (XML) as defined by the Text Encoding Initiative (TEI) [5] *Guidelines for Electronic Text Encoding and Interchange* for representing the structural, renditional, and conceptual features of texts [6]. Once the files are created, they are stored in a database.

The process of transferring the data into machine-readable XML files proves to be a complex task, as textual information (hyperlemmata, lemmata, grammatical information, bibliographic information, etc.) has to be extracted from the original file, marked up properly, and written down into a new file. This task belongs to text mining, an area of computer science "broadly defined as a knowledge-intensive process in which a user interacts with a document collection over time with the help of a suite of analysis tools" [7]. Text mining can be also described as an interdisciplinary field that is based on "information retrieval, data mining, machine learning, statistics, and computational linguistics" [8].

A general architecture of a text mining system consists of four parts: preprocessing; core mining operations; presentation layer components and browsing functionality; refinement techniques (postprocessing) [7]. The present paper describes preprocessing, core text mining operation, and one of postprocessing processes (namely, lemmatization), and leaves the full and detailed description of the whole system to be covered in future publications. Yet, a short description of the system is presented in Section 4 in order to give an idea of the relevant positions of those parts, which are the topic of the present paper.

"Low-quality data will lead to low-quality mining results" [8], therefore, preprocessing includes "all routines, processes, and methods required to prepare data for a text mining system's core knowledge discovery operations" [7]. In our task, preprocessing prepares the original texts of glossaries, transforming them into the form which is best suited for the subsequent core operations. Core text mining operations include "pattern discovery, trend analysis, and incremental knowledge discovery algorithms" [7]. In the present work the core operation is pattern discovery, and it includes not only the initial recognition of text elements, but also the identification of the correct positions of tagged elements (or sets of elements) in an entry.

The glossaries are based on word forms that occur in different manuscripts that in turn originate from different regions. As a result, word forms with the same meaning can have different graphical representations. This also applies to lemma forms used in some of the glossaries. But how can these word forms are allocated to a single superordinate form? The issue can be solved by assigning a normalized form — called "hyperlemma" — to every

Church Slavonic and Greek wordform. The normalized word forms are provided by [9]. This process is called lemmatization and it allows finding all examples of a particular lexeme, independently of their particular inflectional form, as they are used in a text [10]. Although in many cases the hyperlemma is identical to the lemma, these word forms have to be checked against special dictionaries [11 – 16] and could be corrected if necessary. In the present work, lemmatization is done automatically by checking the forms against an electronic lexicon as well as manually by a linguist.

In order to create efficient, robust and easily maintainable text mining software, software design principles have to be used, which are the part of software engineering discipline. The definition, given by the ISO/IEC/IEEE Systems and Software Engineering Vocabulary (SEVOCAB), states that "software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software" [17]. The Guide to the Software Engineering Body of Knowledge (SWEBOK Guide) defines software design as the analysis of software requirements in order to develop a software's internal structure that result in the description of the software architecture components sufficiently detailed to enable their construction [18]. In the current paper we briefly describe the application of some software design principles to the creation of our software.

The present paper is organized as follows. Section 1 describes the preprocessing of the original texts of glossaries. An original algorithm for pattern recognition is presented in Section 2. Section 3 is devoted to the realization of lemmatization. The entries from a Church Slavonic-Greek glossary [19] are used for the explanation and illustration of concepts and features throughout the sections listed above. Examples of application of the software design principles to our software, its main modules, and the general structure of the text mining system are given in Section 4. Finally, future perspectives are discussed in Section 4.

## 1. Preprocessing

In order to avoid time consuming initial digitization of printed material, the already existing digital files with Church Slavonic-Greek glossaries, written in Microsoft Word binary file format (.doc extension), were taken. These files were created around the year 2000 and used as templates for the printed editions.

The .doc format contains a lot of additional formatting, which holds no information about the contents of a glossary itself and can also make a text pattern recognition to be more complicated. Therefore, as a preprocessing step, the documents are converted from the .doc format to the Unicode plain text format (.txt extension), which has very little formatting.

Unfortunately, at the time of creation of the original .doc formatted glossaries, the Unicode standard did not yet include the entire set of Church Slavonic characters, so in order to represent the Greek, Latin and Church Slavonic writing systems, a multitude of non-unicode fonts were used. As long in our case we only use it for printing a book, those fonts don't cause any trouble. Yet, if one intends to work with the text contents of a glossary, the conversion from .doc to .txt file format proves to be impossible, because applying a single Unicode font to the document makes most of the characters unreadable (see Fig. 1).

In order to overcome this problem, a code point of each character in a .doc file needs to be transformed into an appropriate Unicode equivalent first. In the current work this transformation is done semi-automatically by the *SlaVaComp-Konvertierer* — a specially developed (Windows-)software [20].

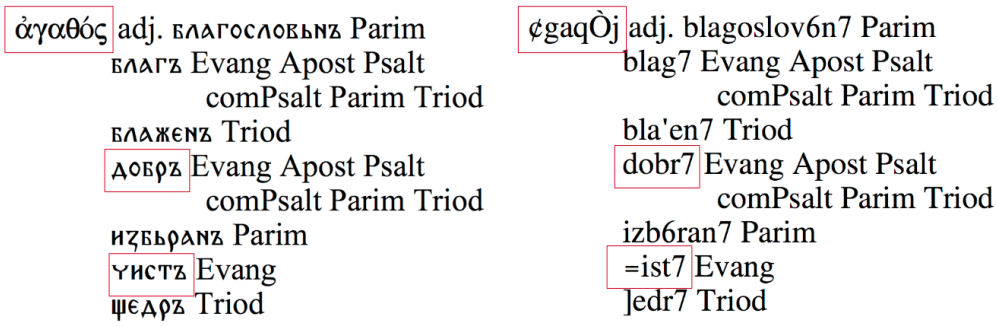


Fig. 1. An exemplary glossary entry using non-Unicode fonts (left) and a single Unicode font (Roman Cyrillic Std., right)

Let us also note, that since starting from the Microsoft Word 2007 the default Microsoft Word binary file format (.doc extension) has been replaced by the Office Open XML format (.docx, .docm extension). The *SlaVaComp-Konvertierer* also converts the original .doc formatted glossaries to the .docx format, allowing their reading by modern computing systems and continuing the usage as templates for the printed editions if necessary.

## 2. Original algorithm for pattern recognition

In the present work the core text mining operation is pattern discovery. It does not only include the initial recognition of text elements, but also the subsequent identification of the correct positions of tagged elements (or sets of elements) in an entry in the TEI XML file. Whereas all entries in glossaries share a general structure, they can significantly differ in details. A first position is always filled by a lemma, other types of elements will follow after: grammatical information, bibliographic information, counterpart(s), editor's comments, etc. These types, in turn, can consist of several subtypes. For instance, grammatical information is further divided into cases, genders, times, etc. Bibliographic information contains numbers of pages and numbers of lines.

Not all types of elements are present in every entry and relative positions of specific information types can also differ. Besides, the glossaries contain a vast amount of special cases, where some elements are written differently, special types of information are encountered, and artifacts whose origins cannot always be explained are present. Thus, the structures of entries, even within one glossary, can be strongly heterogeneous.

Correct XML tagging may require grouping, splitting, shifting, replacing, deleting of original text elements. Thus, element-wise tagging, starting from a first element and going on sequentially, proved to be difficult. We suggest another approach, in which XML tagging is done block-wise according to a so-called map of entry.

The idea is based on one of the author's experience in the field of Computational Fluid Dynamics. One approach to the solution of a wide range of problems on fluid flows in domains with complicated geometries is to use a one-to-one mapping to new coordinates, where a domain becomes simpler (a unit cube, for instance) [21]. While the governing equations become more complicated, their finite-difference approximations can be done far more easier and with a higher accuracy.

This idea can be applied to our current task in the field of Digital Humanities. A type of each element of an entry (hyperlemma (see Section 3), lemma, grammatical, bibliographic

information, counterpart, etc.) is written down into an array, which is named as the "map". There is a one-to-one correspondence between the map elements and the text elements of an entry. By knowing types and positions of text elements, it is easy to perform grouping, splitting, shifting and other actions, and ultimately to mark up the text using the TEI XML. Therefore, we operate in the "space of element types", and the text elements are processed according to their map "images".

Let us present the implementation using the example of an entry from a Church Slavonic-Greek glossary [19] (see Figure 2). The glossary is processed entry-wise, an entry is handled part-wise and can have one or more parts. Since our software was written using Perl programming language and the numbering of elements in Perl arrays starts at 0, the parts of the entry in Figure 2 are denoted as part[0], ... , part[4]. A part contains either a lemma, which is followed by other data, or variant, which is represented by one or more graphical variants and can be preceded by grammatical information and/or followed by other data. Each part is processed separately according to its map. However, as all parts of an entry are stored, the elements can be relocated from one part to another, if needed.

врема сущ. с.ед.
ИП врема 88r8 (καιρός)
П (без) вр҃емене 62v1 (ἀχρόνως), (до) времене 78r12
ВП врѣма 2r3 (καιρόν), (въ) врема 11v15 (въ благо врема – εὐκαίρως), 62v1
мн. ВП времена 1r19 (καιρούς), врѣмена 83v1

Fig. 2. The example from the glossary [19]

```
part[0] of entry[0] = врема сущ. с. ед.

part[0] of entry[0]:
        0th level map = [lemma] [gram_info] [gram_info] [gram_info]

part[0] of entry[0]: 0th level map contains 1 block(s):
        block[0] = [lemma] [gram_info] [gram_info] [gram_info]


block[0]: tag 1st level [gram_info] boxes (assemble 1st level [gram_info] boxes in one box)
        1st level map = [lemma] [gram_info]

part[0] of entry[0]: 1st level map contains 1 block(s):
        block[0] = [lemma] [gram_info]


block[0]: tag [lemma] box
        2nd level map = [lemma]

part[0] of entry[0]: 2nd level map contains 1 block(s):
        block[0] = [lemma]


part[0]: tag 3rd level xml boxes (assemble [var] boxes)
        [lemma] box, no [var] box to assemble

part[0] of entry[0]: 3rd level map contains 1 block(s):
        block[0] = [lemma]
```

Fig. 3. The example from the glossary [19]: processing of part[0]

We construct the map using a set of criteria which can be updated when new types of elements or special cases of already known elements are encountered. As it is seen from Figures 3–4, the elements of the map have self-explaining names. For example, [gram_info] stands for grammatical information, [bibl_info] — bibliographic information, [grc_cit] — Greek citation (counterpart), etc.

After the map of a part is constructed, the positions of Church Slavonic graphical variants are identified. This is needed to split the part into blocks so that each block contains either 1) a lemma with its information, 2) a variant's grammatical information and the first (and may

```
part[4] of entry[0] = мн. ВП времена 1г19 (καιρούς), врѣмена 83v1

part[4] of entry[0]:
        0th level map = [gram_info] [gram_info] [graph_var] [bibl_info] [grc_cit] [graph_var] [bibl_info]

        Church Slavonic variant for Church Slavonic lemma from part[0] is represented
        by 2 Church Slavonic graphical variants with positions [2 5]


part[4] of entry[0]: 0th level map contains 3 block(s):
        block[0] = [gram_info] [gram_info]
        block[1] = [graph_var] [bibl_info] [grc_cit]
        block[2] = [graph_var] [bibl_info]

part[4] of entry[0]: new map contains 2 block(s):
        block[0] = [gram_info] [gram_info] [graph_var] [bibl_info] [grc_cit]
        block[1] = [graph_var] [bibl_info]


block[0]: 0th level map = [gram_info] [gram_info] [graph_var] [bibl_info] [grc_cit]
block[1]: 0th level map = [graph_var] [bibl_info]


block[0]: tag 1st level [gram_info] boxes (assemble 1st level [gram_info] boxes in one box)
        1st level map = [gram_info] [graph_var] [bibl_info] [grc_cit]

block[1]: tag 1st level [gram_info] boxes (assemble 1st level [gram_info] boxes in one box)
        no [gram_info](s) in block[1]

part[4] of entry[0]: 1st level map contains 2 block(s):
        block[0] = [gram_info] [graph_var] [bibl_info] [grc_cit]
        block[1] = [graph_var] [bibl_info]


block[0]: assemble [graph_var] box
        2nd level map = [gram_info] [graph_var]

block[1]: assemble [graph_var] box
        2nd level map = [graph_var]


part[4] of entry[0]: tag 3rd level xml boxes (assemble [var] xml boxes) (assemble blocks [0]-[1] in [var] box)

part[4] of entry[0]: 3rd level map contains 1 block(s):
        block[0] = [var]
```

Fig. 4. The example from the glossary [19]: processing of part[4]

be the only) graphical variant, which represents a variant, 3) other graphical variant (each graphical variant with its information is placed in a separate block). If necessary, the blocks have to be combined so that grammatical information of a variant is not kept in a separate block, but precedes the first (and may be the only) graphical variant (see the new map of part[4] of entry[0] in Figure 4). It is obvious that a block with a lemma can be only present in the first part of an entry. This type of structure is dictated by the TEI XML tagging. If no graphical variants are found in a part, then a lemma is added as a graphical variant. In this case the map is changed and the subsequent part is changed accordingly. A one-to-one correspondence between the map and the part allows that to be done easily.

Let us name this initial map as a "0th level map" and its elements as "0th level boxes". The term "box" has been chosen because it is as if we were covering text elements by the boxes with titles declaring element types. A map is processed block-wise, and each block element-wise. Figures 3–4 illustrate the contents of the next paragraphs.

At first, for each block, according to the "0th level map", the basic elements like lemma, bibliographic information, editor's comments, counterpart(s), and others are tagged and stored. Besides, the presence of homonym superscripts is identified, and the corresponding information is stored for future use. After this step some empty "0th level boxes" can appear due to grouping or deleting of elements. These empty boxes are deleted from the map, and the part, i.e. the array with text elements, is updated accordingly. It is very important

```
<entry xml:id="krysko_codIl-1">
    <note>
        <orig>врема сущ. с. ед. ### ИП врема 88r8 (καιρός) ### РП (без) вр́емене 62v1
            (ἀχρόνως), (до) времене 78r12 ### ВП врѣма 2r3 (καιρόν), (въ) врема 11v15
            (въ благо врема – εὐκαίρως), 62v1 ### мн. ВП времена 1r19 (καιρούς), врѣмена
            83v1</orig>
    </note>
    <form type="hyperlemma" xml:lang="cu">
        <orth>AUTO: врѣма</orth>
    </form>
    <form type="lemma" xml:lang="cu">
        <orth>врема</orth>
        <gramGrp>
            <gram type="pos">sb</gram>
            <gram type="gen">n</gram>
            <gram type="num">sg</gram>
        </gramGrp>
    </form>
    <form type="variant" xml:lang="cu">
        <gramGrp>
            <gram type="case">N</gram>
        </gramGrp>
        <form type="hyperlemma" xml:lang="cu">
            <orth>AUTO: врѣма</orth>
        </form>
        <orth>врема</orth>
        <bibl>
            <biblScope unit="page">88r</biblScope>
            <biblScope unit="line">8</biblScope>
        </bibl>
        <cit type="counterpart" xml:lang="grc">
            <form type="hyperlemma" xml:lang="grc">
                <orth>AUTO: καιρός</orth>
            </form>
```

Fig. 5. The beginning of the TEI XML entry for the example from the glossary [19]

to correctly update the part, because the proper text elements have to be taken for further tagging according to the operations with the map.

Next, the "1st level boxes" with grammatical information are assembled in each block. This can be considered as smaller "0th level boxes" are covered by bigger "1st level boxes". Some TEI XML tags are deleted and some are added. Thus, groups of elements are tagged and stored, a "1st level map" is constructed and the part is updated.

Then, the "lemma box" or the "graphical variant box" is assembled in each block. These are the bigger "2nd level boxes", which cover the sets of "0th level boxes" and "1st level boxes". The groups of elements are tagged and stored, and a "2nd level map" is constructed, where every block either starts with a lemma or contains a graphical variant. The part is updated correspondingly as well.

On the next step, "3rd level boxes", i.e. "variant boxes" are assembled for each block. They contain all graphical variants which represent a variant and the grammatical information relevant to it. The tagged lemmata and variants form a tagged part. And, finally, all tagged parts of an entry are assembled into a tagged entry.

Figure 5 presents the beginning of the TEI XML entry for the example from the glossary [19]. "AUTO" marks the results of the automatic lemmatization which is described below in Section 3.

Let us note, that such pattern recognition algorithm also allows finding corrupt data in the original electronic text sources.

## 3. Lemmatization (postprocessing)

At first, there is no electronic lexicon which could be used for automatic lemmatization. Therefore, a check against special dictionaries [11 – 16] and correction could only be done manually, by a linguist. After the successful transformation of several glossaries into the TEI XML format, a lexicon that contained the hyperlemmata and its associated word forms was created and made available for automatic lemmatization through the XML database. Before a lemma is marked up, it is sent to the database server, and if it is found in the database, the server gives back the respective hyperlemma that is noted accordingly in the TEI XML file. Automatic lemmatization is done simultaneously with the TEI XML tagging, and a copy of lemma is added as "hyperlemma"to the TEI XML file. Let us note that from the implementation point of view automatic lemmatization belongs to the core text mining operation since it is done in the course of pattern recognition (see Figure 7 in Section 4).

A big advantage is the extensibility of the database: with each transformed glossary, new word forms can be added to the database, which in turn helps improving lemmatization. Thanks to this method, it is possible to automatically assign a correct hyperlemma to a great number of headwords, and even though the hyperlemmata still need to be reviewed by a human being, the task is done considerably faster now. Let us call this reviewing operation as a manual lemmatization, emphasizing a human involvement contrary to the automatic lemmatization described above.

In order to make manual lemmatization easy, effective, and less time consuming, a plain text file is created simultaneously with the TEI XML tagging of corresponding word forms. The file has a special structure and provides the word forms for manual lemmatization: hyperlemmata, Church Slavonic lemmata, Greek lemmata, etc. Figure 6 shows the fragment of the structured text file for part[3] (i.e. the 4th line of the text) of the entry from Figure 2. Some other word forms are also listed, for instance, original Greek counterparts, which

16 - 1 - 3 - 1 - 0 graphical variant hyperlemma: AUTO: врѣмѧ
flag = 0
17 - 1 - 3 - 1 - 0 original graphical variant: врѣмѧ
-
18 - 1 - 3 - 3 - 0 Greek counterpart hyperlemma: AUTO: καιρός
flag = 0
19 - 1 - 3 - 3 - 0 Greek counterpart lemma: καιρόν
flag = 0
20 - 1 - 3 - 3 - 0 original Greek counterpart: καιρόν
-

21 - 1 - 3 - 4 - 0 graphical variant hyperlemma: въ
flag = 0
22 - 1 - 3 - 4 - 1 graphical variant hyperlemma: времѧ
flag = 0
23 - 1 - 3 - 4 - 0 original graphical variant: (въ) времѧ
-
24 - 1 - 3 - 6 - 0 original Church Slavonic collocation: въ благо времѧ
-
25 - 1 - 3 - 6 - 0 Greek counterpart (for Church Slavonic collocation) hyperlemma: AUTO: εὔκαιρος
flag = 0
26 - 1 - 3 - 6 - 0 original Greek counterpart for Church Slavonic collocation: εὐκαίρως
-

Fig. 6. The example from the glossary [19]: the fragment (for part[3] of the entry in Figure 2) of the structured text file for manual lemmatization

should not be checked and corrected, but can be useful for a linguist. Besides, additional data for automatic processing of manual corrections is present (the sequences of numbers at the beginning of some lines), which does not hinder the readability of the essential linguistic information.

All lines with word forms, which are the subject of lemmatization, are followed by lines containing a "flag" value (Figure 6). If a linguist finds a wrong word form and corrects it, then a corresponding flag below has to be set to 1 or 2. "1" means a manual correction, "2" means the manual correction of an already automatically corrected word form. The results of the automatic lemmatization are marked as "AUTO" and these changes have to be manually checked in order to improve the electronic lexicon that store the hyperlemmata.

Manual lemmatization has precedence over automatic lemmatization, therefore, specific word form is checked in the database on the server only if it has not been corrected manually. A file with manual corrections is processed by the software, the corrected word forms are noted in the TEI XML file, and a new structured text file is created at each run. See Section 4 for more details on the interaction of the software with input and output data flows.

## 4. Software structure and text mining system

Let us provide a few examples for implementation of the software design principles [18] in our software. The structures of glossaries follow the general principles of the genre. However, they differ considerably in how they represent information. The heterogeneity of the glossaries does not allow processing them in the uniform way. The principle of decomposi-

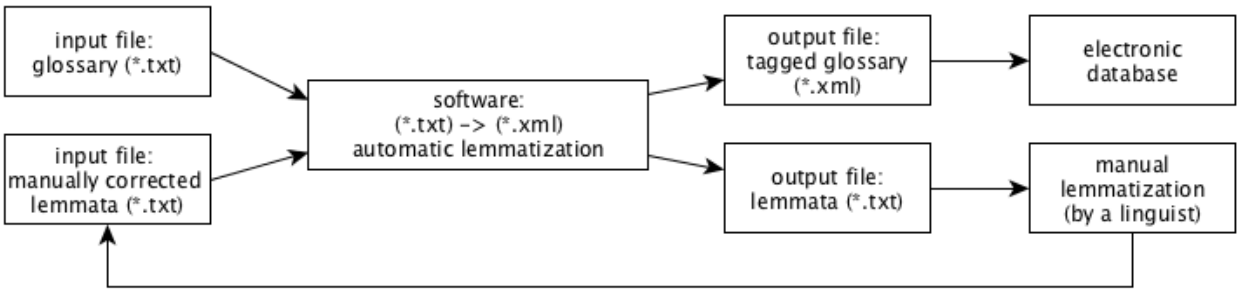


Fig. 7. Text mining system

tion and modularization [18] provides a solution: the software was built from a number of specialized modules with proper interfaces for efficient interactions. Thus, using appropriate sets of modules, the software can process glossaries with various representations for information. Besides, as it was mentioned above, all examined glossaries contain a vast amount of special cases. Taking into account the size of the glossaries, it is extremely time consuming and, thus, practically impossible to find all these special cases manually. They are encountered during the processing of glossaries and lead to changes in pattern definitions or to the introduction of new patterns, and, consequently, to update of the corresponding modules. All modules were made as specialized as possible in order to achieve a low level of coupling ("a measure of the interdependence among modules in a computer program" [17]) and a high level of cohesion ("a measure of the strength of association of the elements within a module" [17]).

Thus, the current version of our software includes the following main modules:

- preprocessing module (general and glossary-specific cleaning, assembling each entry into a text string, i.e. one element of array, formatting);
- pattern recognition module (construction of a map);
- tagging module (mark up using the TEI XML);
- automatic lemmatization module;
- module for processing input data and creating output data for manual lemmatization.

We have developed the original text mining system for processing the glossaries (Figure 7), where computer and human resources interact efficiently. Two plain text files: 1) a glossary, 2) manually (i.e. by a linguist) corrected lemmata are taken as input data. Our software performs two main tasks: 1) transfer of a glossary from the plain text format into the TEI XML format, 2) automatic lemmatization. The output data are 1) a glossary marked up using the TEI XML, 2) a plain text file with lemmata for a linguist to check. The TEI XML file is sent to the electronic database, while the file with the lemmata is processed by a linguist and used in the next run as the input data.

## Conclusion

In this paper we have presented a text mining system for creating electronic glossaries with application to research of Church Slavonic language. In order to enable research of the geographical and chronological differentiation of Church Slavonic language between the 10th and 16th centuries, fifteen Church Slavonic-Greek glossaries had to be marked up using the TEI XML and then transferred into an electronic database.

The digital files with Church Slavonic-Greek glossaries in .doc format were taken as a source. During the preprocessing step, a code point of each character in a .doc file was transformed into its appropriate Unicode equivalent, and the documents were converted from the .doc format to the Unicode plain text format. We have developed an original pattern recognition algorithm for the initial discovery of text elements and the subsequent identification of the positions of tagged elements (or sets of elements) within a TEI XML entry presented. The algorithm was based on the idea from the field of Computational Fluid Dynamics, which was now successfully applied to the problem in the area of Digital Humanities. Postprocessing included automatic and manual lemmatization (assigning a normalized form to every Church Slavonic and Greek lemma). Automatic lemmatization consists in a check against an electronic lexicon and is done in the course of pattern recognition. A linguist has to perform the manual lemmatization by checking a specially structured plain text file.

Due to its universal and modular structure, our text mining system can be used not only for the research of the Church Slavonic language, but it can be applied to processing glossaries in other languages as well.

## Список литературы / References

[1] **Schreibman, S., Siemens, R, Unsworth, J.** A companion to digital humanities. Oxford: Blackwell, 2004. http://www.digitalhumanities.org/companion/

[2] What Is Digital Humanities? http://whatisdigitalhumanities.com

[3] **Fitzpatrick, K.** On scholarly communication and the digital humanities: An Interview with Kathleen Fitzpatrick / By Fred Rowland and Andrew Lopez, January 14, 2015. http://www.inthelibrarywiththeleadpipe.org/2015/on-scholarly-communication-and-the-digital-humanities-an-interview-with-kathleen-fitzpatrick

[4] SlaVaComp (COMPutergestützte Untersuchung von VAriabilität im KirchenSLAvischen). http://www.slavacomp.uni-freiburg.de/

[5] The Text Encoding Initiative (TEI). http://www.tei-c.org/index.xml

[6] The TEI Guidelines for Electronic Text Encoding and Interchange. http://www.tei-c.org/Guidelines/P5/

[7] **Feldman, R., Sanger, J.** The text mining handbook. Cambridge: Cambridge Univ. Press, 2006. 424 p.

[8] **Han, J., Kamber, M., Pei, J.** Data mining: concepts and techniques. Elsevier, 2011. 744 p.

[9] Slovnìk jazyka staroslověnskèho = Lexicon linguae palaeoslovenicae. Vol. I-IV. Praha, 1959–1997.

[10] Supplementary volume dictionaries. An International encyclopedia of lexicography. Supplementary volume: Recent developments with focus on electronic and computational lexicography / R. Gouws, U. Heid, W. Schweickard et al. (Eds.) Berlin, Boston: De Gruyter Mouton, 2013. 1579 p.

[11] **Liddle, H.G., Scott, R., Jones, H.S.** A Greek-English lexicon. Oxford: Clarendon Press, 1996. 2448 p.

[12] **Lampe, G.W.H.** A Patristic Greek lexicon. Oxford: Oxford Univ. Press, 1969. 1616 p.

[13] **Bauer, W.** Griechisch-Deutsches Wörterbuch zu den Schriften des neuen Testaments und der frühchristlichen Literatur. Berlin, New York: Walter De Gruyter, 1998.

[14] **Muraoka, T.** A Greek-English lexicon of the Septuagint. Louvain–Paris-Walpole (MA): Peeters, 2009. 757 p.

[15] **Lust, J., Eynikel, E., Hauspie, K.** A Greek-English lexicon of the Septuagint. Stuttgart: Deutsche Bibelgesellschaft, 1992. 217 p.

[16] **Trapp, E.** Das Lexikon zur byzantinischen Gräzität: besonders des 9.-12. Jahrhunderts. Wien: Verlag der österreichischen Akademie der Wissenschaften, 2001. 316 p.

[17] ISO/IEC/IEEE Systems and Software Engineering Vocabulary (SEVOCAB). `www.computer.org/sevocab/`

[18] Guide to the Software Engineering Body of Knowledge (SWEBOK Guide). `http://www.computer.org/web/swebok`

[19] **Крысько В.Б.** Ильина книга. Рукопись РГАДА, Тип. 131. Лингвистическое издание, подготовка греческого текста, комментарии, словоуказатели. М.: Индрик, 2005. 904 с. **Krys'ko, V.B.** Elias' Book, Manuscript RGADA, Tip. 131. Linguistic edition, processing of the Greek text, comments, glossaries. Moscow: Indrik, 2005. 904 p. (In Russ.)

[20] **Skilevic, S.** SlaVaComp: Konvertierungstool // Slověne = Словѣне. Intern. J. of Slavic Studies. 2013. Vol. 2, No. 2. P. 172–183.

[21] **Khakimzyanov, G.S., Shokina, N.Yu.** Numerical modelling of three-dimensional steady fluid flows on adaptive grids // Russ. J. of Numer. Anal. and Math. Modelling. 2001. Vol. 16, No. 1. P. 33–57.

*Received for publication 26 May 2016*

**Система интеллектуального анализа текста для создания электронных словарей в применении к исследованию церковнославянского языка**

Н. Ю. Шокина, С. Мокен

Вычислительный центр университета Фрайбурга, Германия

Контактный e-mail: `nina.shokina@gmail.com, susanne.mocken@rz.uni-freiburg.de`

В статье представлена система интеллектуального анализа текста для создания электронных словарей в применении к исследованию церковнославянского языка. Для исследования географической и хронологической дифференциации церковнославянского языка в 10–16 веках необходимо было разметить в формате TEI XML и занести в электронную базу данных пятнадцать церковнославянско-греческих словарей. Цифровые файлы со словарями в формате .doc были взяты в качестве исходного материала. На этапе предварительной обработки кодовая точка каждого символа в .doc файле преобразована в свой эквивалент в стандарте Юникод, и документы конвертированы из формата .doc в текстовый формат в Юникод.

Разработан оригинальный алгоритм распознавания образов для обнаружения текстовых элементов и последующего определения позиций размеченных элементов (или наборов элементов) в записи в формате TEI XML. Идея из области вычислительной гидродинамики, лежащая в основе алгоритма, успешно применена для решения задачи в области информационных технологий в гуманитарных науках.

Кратко описано применение принципов проектирования программного обеспечения. Заключительная обработка включала в себя автоматическую и ручную лемматизацию (присвоение нормализованной формы каждой церковнославянской и греческой лемме). Автоматическая лемматизация включает в себя проверку лемм по электронному лексикону и выполняется в процессе распознавания образов для обнаружения текстовых элементов. Ручная лемматизация выполняется лингвистом и заключается в проверке и, при необходимости, корректировке структурированного специальным образом текстового файла.

Благодаря универсальной модульной структуре разработанная система интеллектуального анализа текста может быть применена не только для исследования церковнославянского языка, но и для обработки словарей других языков.

*Ключевые слова*: информационные технологии в гуманитарных науках, интеллектуальный анализ текста, распознавание образов, лемматизация, разработка программного обеспечения, проектирование программного обеспечения, TEI XML, церковнославянский язык.

*Поступила в редакцию 26 мая 2016 г.*