Proceedings of the Federated Conference on Computer Science and Information Systems pp. 181–184 DOI: 10.15439/2017F44 ISSN 2300-5963 ACSIS, Vol. 11

A Method For Data Classification In Slovak Medical Records

Erik Kučera, Oto Haffner and Erich Stark
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava
Bratislava, Slovakia
Email: erik.kucera@stuba.sk

***Abstract*—The topic of representation, classification, and clustering of text documents and information extraction is currently a very researched area. The area of data mining and text mining has its specific problems in the Slovak language. This paper deals with the methods of pre-processing of medical data, namely Slovak health records written in natural language, and their subsequent analysis, especially classification of their parts into classes.**

I. Introduction

DATA mining is a process of analyzing large amounts of data from different perspectives, their summarizing and their use in various sectors. It uses methods of statistics, artificial intelligence, machine learning, mathematics, etc. Data mining is very widely used in practice. For example, in scientific research, for spam blocking, in marketing to decide which customers it is appropriate to send a products offer. Knowledge discovery is a process of (semi-) automatic extraction of knowledge. There are different methods of knowledge discovery but in general, it has following key steps: definition and analysis of our task, obtaining relevant data and its comprehension, data pre-processing, data mining, evaluation and identification of patterns and found knowledge [1] [2].

Data mining process begins with an analysis of the particular task and understanding of existing knowledge and definition of an objective. A process of obtaining relevant data follows. Therefore, it is necessary to decide which attributes in the existing databases are relevant to the task. Then we need to understand this data [3]. It is necessary to decide whether there is sufficient sample for extracting relevant applicable knowledge. All these steps were also applied in our project. In the phase of understanding the problem, we studied the current state of the problem of electronisation of medical records in Slovakia [4]. In the next phase of understanding the data we started studying specific reports, we investigated their quality regarding further processing and subsequent data mining. During the preparation phase, we tried to select a suitable sample of medical records for the next phases of the process. In the phase of modelling the data, we applied various algorithms and tweaked their parameters. Then we evaluated the results achieved by these algorithms.

II. Researched method

In this section, we introduce a methodology, which is a contribution to the field of data mining and categorization of medical records in Slovakia. The result of this procedure is mainly paragraph classification of medical reports and aggregation of data into logical units.

Division of medical records to individual paragraphs - Doctors usually divide logical parts of their documentation into paragraphs. A reasonable step is a division of the original text into these physical units (paragraphs). Each paragraph of the report is saved in a separate text file. We used a library named Apache POI which was used for creating of our application. This application was used for division of documents into paragraphs. Using this application, we can easily divide a lot of documents into distinct paragraphs.

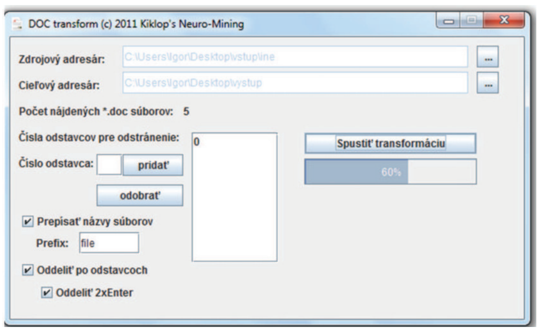
Lemmatization - The next step is a lemmatization of texts using software tool Morphonary. During the process of lemmatization, we ensure the words with the same word root (i.e. doctors and doctor) are identified as the same term. This is important during paragraph classification process.

Analysis of the list of the most common words in the records - Lemmatized texts should be further investigated with the application RapidMiner. Using the list of the most common words in records, we can analyze whether a certain doctor uses its own atypical abbreviations or words. If these words are found, they should be inserted into lemmatizer's dictionary as a new entry. E.g. if the doctor uses an abbreviation 'pcnt' instead of a word 'pacient' it is good to add a pair 'pcnt - pacient' to lemmatizer to replace all 'pcnt' occurrences with 'pacient'.

Completion of lemmatizer - It consists of the aforementioned addition of atypical words to the dictionary of lemmatizer.

Categorization of the paragraphs - This is one of the most important things on our progress in the researched area. In this step, we try to classify paragraphs of medical records to the correct category like prescripted medicine or patient's subjective complaints. In the practical part of the paper, we try to introduce the procedure using RapidMiner how to classify these paragraphs. Since this is a broader issue, our results and the success of each method is given for clarity in a special chapter.

Classification of data into logical units - Using tokenization we divided the text into small pieces, and many of them are related. Therefore, in the last phase, we tried to group each part of the medical reports using logical structures into compact units.


DOC transform (c) 2011 Kiklop's Neuro-Mining

Zdrojový adresár: C:\Users\Igor\Desktop\vstup\ine

Cieľový adresár: C:\Users\Igor\Desktop\vystup

Počet nájdených *.doc súborov: 5

Čísla odstavcov pre odstránenie: 0

Číslo odstavca:

[✓] Prepisať názvy súborov

Prefix: file

[✓] Oddeliť po odstavcoch

[✓] Oddeliť 2xEnter

Fig. 1. Application "DOC transform" programmed using Apache POI library

A. *Division of medical records into paragraphs*

We assume that documents produced by doctors are created by Microsoft Office or OpenOffice. We used the library Apache POI [5] for processing these files. Our application can be used for processing of a large amount of files (health records). We can send the input and output folder where processed documents (divided into paragraphs) are stored (Fig. 1).

B. *Lemmatization*

We used software tool Morphonary for lemmatization because of its best ratio 'speed / efficiency / complexity' [6]. This tool was programmed in Java.

Morphonary works with three dictionaries: Dictionary of foreign words (in Slovak language *Slovník cudzích slov - SCS*), Dictionary of Slovak language (in Slovak language *Slovník slovenského jazyka - SSJ*) and declined words dictionary. SCS contains approximately 60 000 words in basic form. SSJ contains approximately 12 000 words.

The very important is 'declined words dictionary'. This dictionary contains 1730 words in basic form and their inflected forms. These 1730 words are selected to represent the variability of inflected forms of words. There are pairs 'basic form / inflected form'. If the algorithm does not find a word (that is going to be lemmatized) in SSJ or SCS, this process occurs. On the basis of these word pairs, the algorithm evaluates the suffixes. In the case of similarity, the algorithm replaces the word to be lemmatized with its predicted pattern - the root word in the basic form found in this dictionary.

C. *Analysis of the list of the most common words in the records*

This step consists of pre-processing (tokenization, stop-words removal, filtering tokens shorter than 2 characters and longer than 99 characters) and further processing by Rapid-Miner application (Fig. 2).

Now we can use RapidMiner for obtaining of the wordlist and analyze it (Fig. 3).

D. *Completion of lemmatizer*

In health records, there are frequently used foreign words, technical terms or own doctors' abbreviations. By analysis,

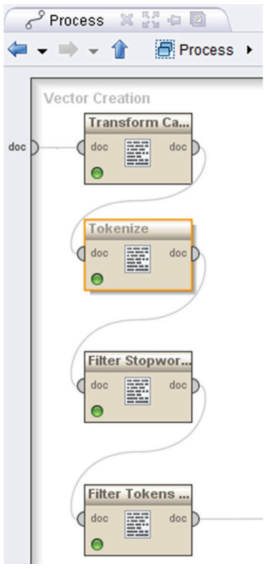
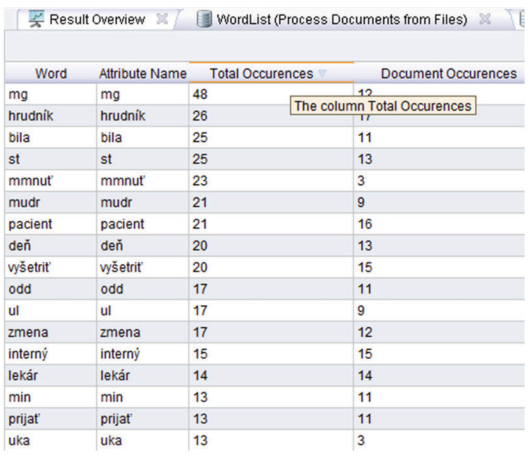


Fig. 2. Pre-processing - RapidMiner application


Result Overview
WordList (Process Documents from Files)
The column Total Occurences

Word	Attribute Name	Total Occurences	Document Occurences
mg	mg	48	12
hrudník	hrudník	26	17
bila	bila	25	11
st	st	25	13
mmnuť	mmnuť	23	3
mudr	mudr	21	9
pacient	pacient	21	16
deň	deň	20	13
vyšetriť	vyšetriť	20	15
odd	odd	17	11
ul	ul	17	9
zmena	zmena	17	12
interný	interný	15	15
lekár	lekár	14	14
min	min	13	11
prijať	prijať	13	11
uka	uka	13	3

Fig. 3. Wordlist - RapidMiner application

it was investigated that lemmatizer has the lowest percentage of correctly processed words with this group of tokens. We decided to find a method that does not lemmatize these words. If a word is an abbreviation, it should be transformed into its full form.

This is a brief description of proposed method:

1) For several randomly chosen health records of a certain doctor, we obtain a word list that belongs to mentioned word group.
2) A list of these words is shown to the physician who translates it into the full form.
3) We export these pairs of words ('abbreviation - full form') to CSV file.
4) We import this file in the Morphonary and add it to the declined words dictionary.
5) After lemmatization process, these abbreviations and special words will be translated into their full forms.

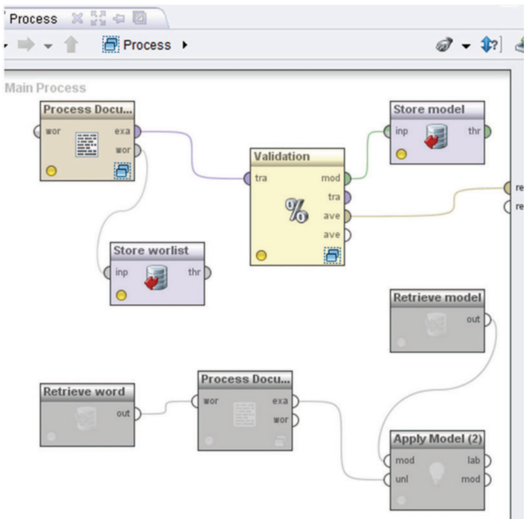


Fig. 4. Classification - making of classifier

E. Categorization of paragraphs

For categorization of paragraphs, we mainly use classification (supervised learning). That means that we have to classify some sample data manually. Then a classifier is created, and then this classifier is used to classify the other data. We can easily perform this process in Rapidminer. We use mainly three methods of classification - Naive Bayes, k-nearest neighbour (k-NN) and decision tree.

We will briefly show how to use k-NN method in RapidMiner (using of Naive Bayes or decision tree is very similar).

The basis is to correctly set the operator named *Process Documents from Files*. We can see the entire process in Fig. 4.

In a window, we have 2 groups of operators (Fig. 4). The lower group is deactivated in the figure. Next, we set input files for operator *Process Documents from Files*. These input files are a training set of data that are manually classified by a supervisor (human). Our categories are epicrisis, code of the medical facility, medicine (or therapy), names (of doctors, etc.), subjective problems, examination, and conclusion.

We need also the operator named *X-Validation* that can be found in operators menu: *Evaluation - Validation - X-Validation*. This operator is used for teaching the classifier and its cross-validation.

When we double-click on the operator *X-Validation* we can see the window of cross-validation subprocess. This window contains two subwindows: *Training* and *Testing*. We insert the operator *k-NN* (or *Naive Bayes* or *Decision tree*) to subwindow *Training*. Then we set *Measure types* to *NumericalMeasures* and *CosineSimilarity*.

Next we insert operators *Apply Model* (*Modeling - Model Application - Apply Model*) and *Performance* (*Evaluation - Performance Management - Performance*) to subwindow *Testing*. Finally, we link the operators.

We need to add two operators of type *Store* (*Repository Access - Store*) to the main process. We link the first one to the output named *wor* of the operator *Process Documents from Files*. On its settings, we choose the path where the input should be stored. This output is the wordlist from operator *Process Documents from Files*. We will need this output later during classification. The second operator *Store* is linked to the output *mod* of the operator *X-Validation*. By this, we save our trained classifier to a file. We will need it during the classification process.

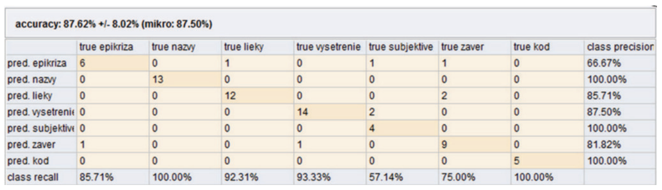
Then we can run the process. Then we open a results window and the tab *PerformanceVector*.

In Fig. 5 there are results of cross-validation that was performed using training data. A sum of values in a row determines the number of documents that are categorized to the certain class. For example, there are 9 files categorized to the category *epicrisis*. A sum of values in a column determines the number of documents that are actually of a certain type/class (we have this information because training data was manually categorized). For example, 7 files really belong to the category *epicrisis*.

The last column *class precision* determines what ratio of documents classified by the operator as epicrisis actually belongs to this class. The last row *class recall* determines what ratio of documents that are actually of a certain class has been really classified to correct class.

The result of cross-validation gives us information about the precision of classifier. We can further modify our training data or parameters of the operator.

Next, we are going to classify uncategorized data by trained classifier. Operators that were used previously should be deactivated. We add new operator *Process Documents from Files*.


accuracy: 87.62% +/- 8.02% (mikro: 87.50%)

	true epikriza	true nazvy	true lieky	true vysetrenie	true subjektive	true zaver	true kod	class precisior
pred. epikriza	6	0	1	0	1	1	0	66.67%
pred. nazvy	0	13	0	0	0	0	0	100.00%
pred. lieky	0	0	12	0	0	2	0	85.71%
pred. vysetreni	0	0	0	14	2	0	0	87.50%
pred. subjektiv	0	0	0	0	4	0	0	100.00%
pred. zaver	1	0	0	1	0	9	0	81.82%
pred. kod	0	0	0	0	0	0	5	100.00%
class recall	85.71%	100.00%	92.31%	93.33%	57.14%	75.00%	100.00%	

Fig. 5. Report of cross-validation

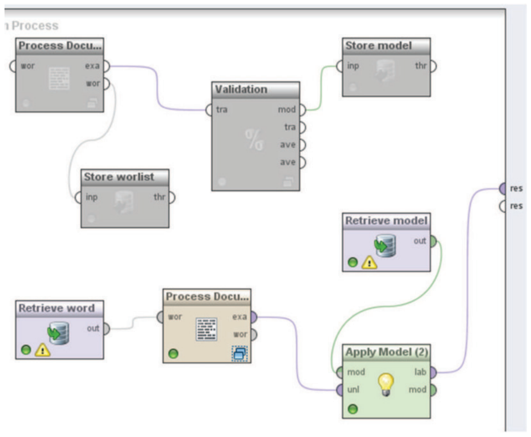
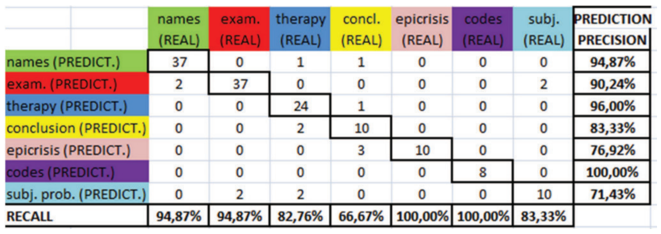
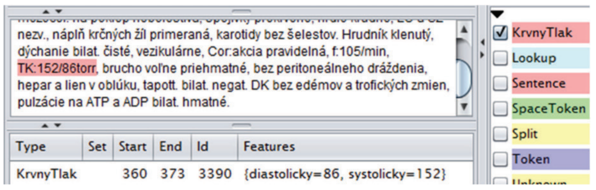


Fig. 6. Classification using trained classifier



	names (REAL)	exam. (REAL)	therapy (REAL)	concl. (REAL)	epicrisis (REAL)	codes (REAL)	subj. (REAL)	PREDICTION PRECISION
names (PREDICT.)	37	0	1	1	0	0	0	94,87%
exam. (PREDICT.)	2	37	0	0	0	0	2	90,24%
therapy (PREDICT.)	0	0	24	1	0	0	0	96,00%
conclusion (PREDICT.)	0	0	2	10	0	0	0	83,33%
epicrisis (PREDICT.)	0	0	0	3	10	0	0	76,92%
codes (PREDICT.)	0	0	0	0	0	8	0	100,00%
subj. prob. (PREDICT.)	0	2	2	0	0	0	10	71,43%
RECALL	94,87%	94,87%	82,76%	66,67%	100,00%	100,00%	83,33%	

Fig. 8. Results of Naive Bayes classifier



nezv., náplň krčných žíl primeraná, karotidy bez šelestov. Hrudník klenutý, dýchanie bilat. čisté, vezikulárne, Cor:akcia pravidelná, f:105/min, TK:152/86torr, brucho voľne priehmatné, bez peritoneálneho dráždenia, hepar a lien v oblúku, tapott. bilat. negat. DK bez edémov a trofických zmien, pulzácie na ATP a ADP bilat. hmatné.

Type	Set	Start	End	Id	Features
KrvnyTlak		360	373	3390	{diastolicky=86, systolicky=152}

Fig. 7. Describing of group of tokens - blood pressure

In settings windows, we choose the folder with documents to be classified. We need also the operator *Apply Model* and two operators *Retrieve* (*Repository Access - Retrieve*). The first one we link to the input *wor* of the operator *Process Documents from Files* and in its settings, we choose a path to the saved wordlist that has been saved during classifier training. The second operator *Retrieve* we link to the input *mod* of the operator *Apply model* and in its setting, we set the path to the model that has been saved during the training process. We can see the entire process in Fig. 6. After execution of the process, we can choose the tab *ExampleSet - Data View* and see the list of classified files.

F. Classification of data into logical units

In the GATE framework, we used *Annotation schema* to describe groups of tokens, and we would like to ensure that this activity has been gradually automated by rules and grammar JAPE (Fig. 7) [7]. This point will help future generations in their next work.

III. Results of Classification

In this section, we briefly describe the results of categorization of paragraphs from health records. We have randomly chosen 25 records that were divided into paragraphs using application created with Apache POI library. We have manually categorized these paragraphs into 7 categories: epicrisis, code of the facility, medicine (or therapy), names (of doctors, etc.), subjective problems, examination, and conclusion.

We describe the results of classification with a supervisor. It was necessary to divide the data (25 health record divided into paragraphs) into training and testing set. The amount of data may seem small, but for the purpose of this project, it is fully sufficient because the goal is to find a suitable methodology and not to test large amounts of data. We afford to say that, despite this, the amount of our data is sufficient also for practice, since it is not possible to create a versatile classifier for any health facility (at least at this stage of the project). It is, therefore, necessary that a particular health facility should prepare its training data, the quantity of which need not to be much larger than in our case. Consequently, these data will be used for the creation of the classifier for the facility.

Training data / Testing data:

1) 13 / 39 paragraphs of class *names (of doctors, etc.)*
2) 15 / 39 paragraphs of class *examination*
3) 13 / 29 paragraphs of class *medicine (or therapy)*
4) 12 / 15 paragraphs of class *conclusion*
5) 7 / 10 paragraphs of class *epicrisis*
6) 5 / 8 paragraphs of class *code of the medical facility*
7) 7 / 12 paragraphs of class *subjective problems*

We achieved the best results with Naive Bayes classifier. The results of Naive Bayes classifier are very good (Fig. 8). There is a certain error in categorizing of paragraphs of type *conclusion* but in practice this class is interchangeable with *epicrisis* or *therapy*. We can see that this is where the classifier categorized the remaining entries so this is not a big error.

IV. Conclusion

This paper deals with the methods of pre-processing of medical data, namely Slovak health records written in natural language, and their subsequent analysis, especially classification of their parts into classes. We tried to achieve progress in the field of text mining in health records. We researched a successful method of paragraphs classification using RapidMiner. The best results were achieved using Naive Bayes classifier. It will be a challenge to try this method for health records from different medical facilities.

Acknowledgment

This work has been supported by the Cultural and Educational Grant Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic, KEGA 030STU-4/2015 and KEGA 030STU-4/2017, by the Scientific Grant Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic under the grant VEGA 1/0819/17.

References

[1] J. Paralič, *Knowledge discovery in texts (in Slovak)*. Kosice: Equilibria, sro, 2010. ISBN 978-80-89284-62-7
[2] P. Berka, *Data mining in databases (in Czech)*. Praha: Academia, 2003. ISBN 80-200-1062-9
[3] J. Paralič, *Knowledge discovery in databases (in Slovak)*. Elfa, 2003.
[4] S. Balogh, F. Lehocki, D. Ivaniš, E. Kučera, M. Lajtman, and I. Miňo, "Data processing from mhealth patient data acquisition related to extracting structured data from eh records," in *International Conference on Wireless Mobile Communication and Healthcare*. Springer, 2012, pp. 255–262.
[5] T. A. P. Project. (2011) Apache poi - text extraction. [Online]. Available: http://poi.apache.org/text-extraction.html
[6] R. Novotný and S. Krajci, *Lemmatization of Slovak words by a tool Morphonary*. Vydavatelstvo STU, 2007.
[7] Xml and More. (2011) Java annotation patterns engine (jape). [Online]. Available: http://xmlandmore.blogspot.sk/2011/05/java-annotation-patterns-engine-jape.html