APPLICATION OF BOOTSTRAP TECHNIQUES FOR POLICE SUMMARIES RETRIEVAL

Virginijus Kaušas[1], **Tomas Krilavičius**[2, 3], **Žygimantas Medelis**[1], **Danas Zuokas**[1, 4]
[1]UAB TokenMill, Naugarduko 56-25, Vilnius, Lithuania
[2]Faculty of Informatics, Vytautas Magnus University, Vileikos 8, Kaunas, Lithuania
[3]Baltic Institute of Advanced Technology, Vilnius, Lithuania
[4]Faculty of Mathematics and Informatics, Vilnius University, Naugarduko 24, Vilnius, Lithuania
E-mails: virginijus.kausas@tokenmill.lt; t.krilavicius@gmail.com; zygimantas.medelis@tokenmill.lt; danas.zuokas@gmail.com

Abstract. The amount of information that is created, used or stored is growing exponentially and types of data sources are diverse. Most of it is available as an unstructured text. Classical Information Technology techniques are not sufficient to process it and exploit its full potential. Information Retrieval (IR) and Natural Language Processing (NLP) provide a number of instruments for information analysis and retrieval. In this paper we present a novel application of NLP for Lithuanian language. We demonstrate that a combination of IR and NLP tools with appropriate changes are successfully applied to Lithuanian texts.

Keywords: Information Retrieval; Natural Language Processing; Named Entity Recognition; Machine Learning; Stemming; Soundex; *n*-grams.

1. Introduction

The amount of information and types of data sources used is growing exponentially. With proliferation of Internet and widespread use of text processing majority of content is available in an unstructured text form. This leads to huge and heterogeneous volumes of accessible data. Yet the traditional IT techniques are ill suited to effectively analyze this type of information.

Information Retrieval (IR) and Natural Language Processing (NLP) provide a set of alternative instruments for information retrieval and analysis (Baeza-Yates, Ribeiro-Neto 1999; Manning, Raghavan, Schutze 2008). Unfortunately, most of NLP techniques are developed only for English and other popular languages (e.g., French, Spanish), while solutions for Lithuanian language are missing. Lithuanian is a highly inflected language, e.g. declension for nouns and adjectives. Due to such complexity and a lack of (freely) available resources like WordNet (Miller 2009) or annotated corpora, these techniques are not directly applicable to Lithuanian texts.

In this paper we present an application of NLP for Lithuanian language that shows how a combination of existing IR and NLP tools with appropriate changes can be successfully applied to Lithuanian text. We demonstrate by applying a bootstrap methodology for retrieval of country event summaries provided by Lithuanian Police. A two step procedure is used to identify named entities (NER) such as places and persons. We use GATE/JAPE (Gate Development Team. 2010) for rule based annotation of text, correct errors manually (if necessary), and then use annotated text to train Conditional Random Fields (Lafferty, J.; McCallum, A.; Pereira, F. 2001) and use them to NER-annotate documents. Documents are grouped by their theme as well using unsupervised and supervised machine learning algorithms. Then an enhanced index is built, where for each term its stem and NER annotations are provided. Search is performed on stems and annotation information provides faceted search facilities. Demo version of the system is available at http://policija.tokenmill.lt/.

The paper has the following structure. In Section 2 we introduce the document corpus, Section 3 describes text-preprocessing tasks, Section 4 talks about NER annotation solution, while document grouping is described in Section 5. A short introduction to demo environment of the system is given in Section 6.

2. Data: Lithuanian Police country events summaries

We run our application on the corpus of 14305 daily event reports from January 1st, 2007 to June 30th, 2010 provided by the Lithuanian Police. The reports are available at http://www.policija.lt/lt/salies-ivikiu-suvestine. A typical report is about 25-75 words long and is quite a structured text, for example:

> Gegužės 13 d. 8 val. 45 min. Ukmergės r., kelyje Vilnius - Panevėžys 70-ame km, motociklas "Kawasaki ZX-6R", vairuojamas neturinčio teisės vairuoti tokio galingumo motociklą Vilniaus apskrities vyriausiojo policijos komisariato Patrulių rinktinės 6-osios kuopos 1-ojo būrio patrulio J. Č. (gimęs 1981 m.), atsitrenkė į priešpriešinio eismo juostos atitvarus. Vairuotojas sunkios būklės paguldytas į ligoninę.[1]

The main topics of the reports include auto accidents, thefts, assaults, fires, bribes and other.

3. Text Preprocessing

In order to perform a meaningful text analysis documents must be preprocessed. The main tasks of preprocessing are spelling and phonetic translation, error correction and word form reduction. The system should not only be able to preprocess the corpus documents but also correct errors or understand reduced word forms in the search queries.

3.1. Data normalization

Textual information comes in various degrees of quality: from grammatically correct well formulated, punctuated and spelled - as most often found in official documents or professional articles - to texts that are full of spelling errors or lack diacritic signs. The most prominent example of such text is article's comment sections. If such texts are to be analyzed the errors must be corrected.
There are two main types of errors: spelling and phonetic translation. The first is characterized by incorrect misplaced letters in the word resulting in words, which are not found in dictionary. While the latter occurs when foreign names are phonetically translated into other language.
Spelling errors can be successfully fixed using edit distance algorithms. In our case we use Levenshtein edit distance algorithm which calculates the distance between two character sequences based on the amount of a single character alterations: insertion, deletion, or substitution needed to make those sequences match (Manning, Raghavan, Schutze 2008). Thus the Levenshtein distance between “broken” and “br*a*ken” would be 1 for only one alteration of type substitution need to be done to change “a” to “o”. See (Manning, Raghavan, Schutze 2008) for the details.
Although spelling corrector, based on edit distance algorithms, would solve majority of problems it is incapable of correcting errors made when foreign names are phonetically translated into a given language. For example *Renault* could be turned into Lithuanian equivalent *Reno*. Thus data analysis based on vehicle name extraction would fail, because *Renault* and *Reno* would not be recognized as the same name. Usually, *name-matching* algorithms are used to find matches. Most common algorithms belong to phonetic algorithms group, such as Soundex, Metaphone and Phonex, but lately, *n*-gram based algorithms show quite good results. See (Lait, Randell 1996; Manning, Ch.; Raghavan, P.; Schutze, H. 2008; Holmes, McCabe 2002) for the detailed presentation of the above mentioned algorithms. Usually, phonetic algorithms map all the names to a code, that is the same or very similar to a similarly sounding words, e.g. Soundex maps both, *Vilkaviškis* and *Vilkaviskyje* to *V421*, where first symbol of the code is the same as the first symbol of the word, and the following vowels are omitted and consonants are mapped to their respective groups. Different phonetic algorithms use different mappings, some of them use combinations of letters, not just consonants. *n*-gram based algorithms measure similarity of words by counting number of similar *n*-grams in the words, where *n*-grams are n-letter subsequences of the given word. E.g., in the case of *Vilkaviškis* and *Vilkaviskyje,* corresponding 3-grams would be *{vil, ilk, kav, avi, vis, ski, kis}* and *{vil, ilk, kav, avi, vis, sky, kyj, yje}*. Notice, that we have transformed capital letters to lowercase and removed diacritic symbols, in this case caron from *š*. Then, we can use a selected similarity coefficient, e.g. Pfeifers (Holmes, McCabe 2002) to compare the words:

$$\delta(\text{Vilkaviškis, Vilkaviskyje}) = \frac{\left|\{\text{vil, ilk, kav, avi, vis, ski, kis}\} \cap \{\text{vil, ilk, kav, avi, vis, sky, kyj, yje}\}\right|}{\left|\{\text{vil, ilk, kav, avi, vis, ski, kis}\} \cup \{\text{vil, ilk, kav, avi, vis, sky, kyj, yje}\}\right|} = \frac{\left|\{\text{vil, ilk, kav, avi, vis}\}\right|}{\left|\{\text{vil, ilk, kav, avi, vis, ski, kis, sky, kyj, yje}\}\right|} = \frac{5}{10} = 0.5. \tag{1}$$

Then, *m*-most similar words can be selected. Soundex for Lithuanian language (Krilavičius, Kuliešienė 2010) is

1 At 8.45 AM of May 13th on the 70th kilometer of the Vilnius - Panevezys road in the Ukmerge district a motorcycle "Kawasaki ZX-6R" driven by J. C. (born in the year 1981), who had no right to drive a motorcycle of that power, hit the barriers of the opposite road side. The driver was hospitalized with the severe condition.

available at http://ltsoundex.sourceforge.net/.
In the case police summaries, texts are well formed and grammatically correct. Nevertheless, it does not follow that analytical tools should not perform data normalization. The queries against summaries corpora can themselves be faulty. For example query *sautuvas* (en: gun) will fail, because texts contain only *šautuvas* (with diacritic sign). The same applies to query *reno* while the corpus contains only *renault*. Above-mentioned retrieval strategies can be combined to improve recall (Holmes, McCabe 2002), however we leave it for a future research.

3.2. Error correction

After a set of words is identified as sufficiently similar, one of them has to be chosen as representing the correct form. For example if words: *Varšuva, Warszawa, Varšava* are identified as representing the same entity, one of those three words has to be chosen as correct one and other words have to be changed to this proper form.
We apply two methods to choose the right form of the erroneous word based on use frequency and dictionary. The first method calculates word frequencies in a given corpora and the most often used word is proposed as the canonical one. This has the consequence that not necessarily grammatically correct forms will be proposed. If a word in a given corpora is mostly used in its incorrect form, like *Varšava* then it will be proposed as the base for correction. Dictionary based methods on the other hand do not have this problem because the correct form is chosen from the list of grammatically correct words in the dictionary. Although dictionary based method is preferred, dictionaries are available only for a limited set of domains. It might be feasible to have a dictionary for geographical names, but one might have difficulties forming dictionary of company names for example.

3.3. Word form reduction

Lithuanian language (the same, though to a lesser degree applies to English) possesses the problem of word forms, which have to be recognized as describing the same object. Let us say that police summaries are queried for all assaults involving knifes. The needed query would be “peilis” (en: knife), yet this exact word is never used in the text. Instead it comes in phrase “injured with the knife”, where the modifier “with” in Lithuanian language is not used, instead the ending of “knife” changes to “sužalojo peil*iu*”. Thus the query “peilis” would fail to match “peiliu”.
To solve this problem word forms have to be reduced and only the stem of the word used in the analysis. In the case described above, it would not be the word “peilis” which would be used for querying but only its stem - “peil”. To find stems of Lithuanian words we use modified Porter stemming algorithm (Porter, 1980; Krilavičius, Medelis, 2010).

4. Named Entity Recognition

Processed text can further be analyzed using various NLP techniques. One of the main NLP tasks is Named Entity Recognition (Nadeau, Sekine 2007), which seeks to assign a certain tag or label (to annotate) to a word or group of words. Usually names of persons, organizations, locations, dates or quantities are annotated. In the case of Police reports we are interested in address, vehicle name and tool of injury as well. Thus in the previously given example report: “At 8.45 AM of May 13th” would be identified as date, “Vilnius - Panevezys” as road name, “Ukmerge” as district name, “Kawasaki ZX-6R” as vehicle, “J.C.” as person and so on.
As in our case reports are quite structured we are able to annotate words using (linguistic) rules. Nevertheless, we cannot achieve 100% accuracy, as we are not able to define all rules. In addition the reports may change and old rules may become inappropriate. Thus we need annotation methods, which are not tied up to specific text structure or certain list of frequent words - machine learning based methods.
These two approaches should enhance each other and annotation process should go in cycles to increase precision. We extract information to learn to annotate in order to extract new information and so on. We call such technique *bootstrap*.

4.1. Rule-based entity detection

We use open source text engineering software GATE and its JAPE grammar (Gate Development Team, 2010) for rule-based entity detection. We improve and extend rules set provided in (Kapočiūtė, Raškinis 2005) for Lithuanian text, taking into consideration text specifics as well. We create a GATE application consisting of the following steps:

1. Identify Tokens - token in this case is any character sequence between two SpaceTokens;
2. Run a set of JAPE rules identifying entities: persons, vehicles, injuries, dates, locations, etc.;
3. Run group assignment JAPE rule to categorize documents.
4.

As a result we have documents with annotations attached to tokens (words). JAPE rules describe the conditions, which have to be satisfied for a token to receive an annotation. For example injury rule is defined as follows:

1. We look for a word from a list which indicates injury tool (knife, ax, gun etc.), presence of this pattern is optional;
2. Then we look for a word from a list which indicates body part which is injured (head, chest, hand etc.), presence of this pattern is optional;
3. Then again we look for an optional word describing the severity of the injury;
4. The whole rule is then completed by the required word indicating action ("injured") and earlier identified person.

Already from this example we can see the limitations of rule-based annotation: first of all we need lists of words, which of course cannot be complete (we do not know all possible tools of injury); and second, rules cover only words in a certain sequence.
After running GATE application we have annotated approximately 400000 out of 900000 tokens i. e. a half. As a special interest we have found 865 injury tools where 574 of them are knifes. But if we perform a simple search with the key phrase "injured with the knife" we get 766 matches. This shows that this rule does not cover a considerable number of cases. Of course we could then extend the rule. Alternatively, we could use machine-learning techniques to decrease number of errors. It is described in the next section.

4.2. Machine learning entity detection

Results of the rule-based named entity annotation can further serve as an input for the machine learning named entity annotation. The essence of supervised machine learning is for each observation with a set of features to predict the outcome of the variable of interest (Nadeau, Sekine 2007). Usually, supervised machine learning based NER consists of the following steps:
1. Prepare data for training - the data for which the values of the feature and outcome variables are known;
2. Using probabilistic methods establish functional relationship between features and outcome (determine feature weights);
3. Using feature weights predict the value of the outcome variable.

We use Conditional Random Fields (CRF) (Lafferty, McCallum, Pereira 2001) as a probabilistic model for token categorization. There are several CRF parameter estimation implementations: CRF++, FlexCRFs, MALLET, crfsuite; we use the latter.
In our case the variable of interest is a class of a token (injury tool). We consider only very simple features - surrounding tokens (features F01, F02, F03) and sequences of two tokens (feature F04, F05). The following table is a snippet of training data for a sequence of tokens "kilus ginčui, peiliu sužalojo J. T." (en: in the event of dispute injured with the knife J. T.):

Table 1. The snippet of a train data.

not-tool	F01=,	F02=kilus	F03=ginčui	F04=,/kilus	F05=kilus/ginčui
not-tool	F01=kilus	F02=ginčui	F03=,	F04=kilus/ginčui	F05=ginčui/,
not-tool	F01=ginčui	F02=,	F03=peiliu	F04=ginčui/,	F05=,/peiliu
tool	F01=,	F02=peiliu	F03=sužalojo	F04=,/peiliu	F05=peiliu/sužalojo
not-tool	F01=peiliu	F02=sužalojo	F03=J	F04=peiliu/sužalojo	F05=sužalojo/J
not-tool	F01=sužalojo	F02=J	F03=.	F04=sužalojo/J	F05=J/.
not-tool	F01=J	F02=.	F03=T	F04=J/.	F05=./T
not-tool	F01=.	F02=T	F03=.	F04=./T	F05=T/.
not-tool	F01=T	F02=.	F03=(	F04=T/.	F05=./(

Finding informative features is very important. On the one hand more complex features could increase the precision of the prediction, on the other hand it could increase computational complexity. Choosing features, which occur at least some defined value, is a good method to filter out unimportant features. In our case simple features occurring at least 10 times suffice to obtain quite reasonable results. Following table gives features with the biggest weights, i. e. that are most important for the prediction of a token class:

Table 2. Features with the biggest weights.

F01=ją *not-tool* (en: her)	F03=peiliu *not-tool* (en: with a knife)	F03=sužalojo *tool* (en: injured)	F04=ją/peiliu *not-tool* (en: her/with a knife)	F01=kuris *not-tool* (en: which)	F01=namuose *not-tool* (en: at home)	F03=į *tool* (en: at)	F01=peiliu *not-tool* (en: with a knife)
7.281526	7.173412	6.865895	6.602659	6.165444	5.457446	5.248100	5.204757

E. g., the word “sužalojo” (en: injured) gives the highest weight that the previous token belongs to class *tool*. While “peiliu” (en: with the knife) gives very high weight that the next token belongs to class *not-tool*.
After training the model with the chosen features we predict token class on the same training data. We do this because it is more important for us to fix erroneous rule-based annotations rather than to annotate unseen documents. Inspecting the results shows that we are able to correct token annotation. We get 77 cases involving injury tool knife where rule-based annotation assigned class *not-tool*, but machine learning annotation assigned class *tool*. On the other hand we get 28 cases where machine-learning annotation assigned class *not-tool*, in contrast to the correct rule-based annotation. Summing up, we obtain 49 improved (corrected) cases.
Another important result is that rule-based annotation identifies only two injury tools: ax and knife, but there are other injury tools like hoe or screwdriver. Applying CRF we get probability around 0.2-0.3 for these tokens to belong to *tool* class. Threshold to classify a token is 0.5 and formally these do not classify as tools, but it shows the potential of machine learning techniques.
Machine learning annotation reveals new facts which were previously unknown. This allows extending rules for a second round rule-based annotation. That is a cyclical process, repeated until the desired precision and recall are achieved.

5. Grouping

Sections 3 and 4 present basic NLP tasks at a level of token. In this section we demonstrate two major NLP tasks at a document level: clustering and classification (Baeza-Yates, Ribeiro-Neto 1999; Manning, Raghavan, Schutze 2008). The former being an example of unsupervised and the latter - supervised machine learning method. Both are used when grouping documents.

5.1. Document representation

Both, clustering and classification are very well elaborated techniques on numeric data. In order to use them first textual data must be transformed into numeric. We use classical document representation - document-term matrix where each row represents one document and each column - term. There are several ways of filling this matrix from very simple - binary, which gives value 1 if a given term occurs in a given document and 0 otherwise to more complex. Most common way is the so called tf-idf (term frequency-inverse document frequency) that calculates an estimate for a term *i* and document *j* (Baeza-Yates, Ribeiro-Neto 1999; Manning, Raghavan, Schutze 2008):

$$(tfidf)_{i,j} = tf_{i,j} \times idf_i, \quad tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}, \quad idf_i = \log\left(\frac{|D|}{|\{d : t_i \in d\}|}\right), \tag{2}$$

where $n_{i,j}$ is the number of occurrences of the considered term t_i in document d_j, and the denominator is the sum of number of occurrences of all terms in document d_j, that is, the size of the document $|d_j|$; $|D|$: cardinality of *D*, or the total number of documents in the corpus $|d_j|$, $|\{d : t_i \in d\}|$: number of documents where the term t_i appears[2]. A high weight in *tfidf*, i. e. high importance of the term, is reached by a high term_frequency (in the given document) and a low document frequency of the term in the whole collection of documents. Hence, the weights tend to filter out terms that repeat in all documents. Using such techniques we get a document represented by a vector in *d*-dimensional space of terms.
In order to obtain more meaningful results non-informative words, or so-called *stop words* (conjunctions, pronouns etc.), are removed from the documents and words are stemmed. Such preprocessing minimizes the term space dimensions as well.

2 http://en.wikipedia.org/wiki/Tf-idf

5.2. Clustering: unsupervised grouping

In this subsection we present the results of document clustering for Lithuanian Police daily reports. Each report is treated as a separate document. Together with tokens themselves we use sequences of two tokens - 2-grams as terms. In order to diminish the dimension of term space we consider only terms that occur at least 100 times, in at least 100 documents but at most in 10 % off all documents.
Although there are numerous clustering methods, we use a classical k-means method (Baeza-Yates, Ribeiro-Neto 1999; Manning, Raghavan, Schutze 2008), because it shows good results and is well tested in practice. We employ implementation provided with open source software Apache Mahout (Mahout Development Team, 2010), which is used due to its scalability and readiness to work with textual data. Next table shows most frequent terms of 10 clusters determined by k-means method.

Table 3. Most frequent terms of 10 clusters.

Cluster 1	pastebėta en: detected	nuostolis en: damage	litų en: litas	pavogta en: stolen	nuostolis litų en: damage in litas
Cluster 2	mirtinai en: lethal	partrenkė en: hit	mirtinai sužalojo en: deadly inured	sužalojo en: injred	automobilis en: vehicle
Cluster 3	rastas en: found	areštinę en: in custody	lavonas en: corpse	uždarytas en: locked	uždarytas areštinę locked in custody
Cluster 4	ligoninę en: to hospital	pristatytas en: presented to	paguldytas en: admitted to	aplinkybės en: circumstances	aplinkybės aiškinamos en: circumstances being cleared
Cluster 5	neblaivus en: drunk	prom en: part per million	kyšį en: bribe	išvengti en: avoid	pažeidimą en: violation
Cluster 6	sužalojo en: injured	peiliu sužalojo en: injured with a knife	žaizdos en: wounds	paguldytas ligoninę en: hospitalized	peiliu en: with a knife
Cluster 7	eismo en: traffic	įvykio metu en: during event	eismo įvykio en: traffic accident	įvykio en: event	vairuojamas en: driven by
Cluster 8	materialinis nuostolis en: material damage	materialinis en: material	pagrobė en: rob	ryšio en: communication	nuostolis en: damage
Cluster 9	kreipėsi en: applied	pareiškė en: declared	kuris en: which	dieną en: day	pačią dieną en: same day
Cluster 10	gaisras en: fire	kilo en: strated	kurio metu en: during which	kilo gaisras en: fire started	gaisras kurio en: fire during which

In such a way we can assign a topic to each cluster. For example Cluster 1 can be identified as 'theft' cluster, Cluster 2 - 'lethal traffic accident', Cluster 5 - 'bribe' or Cluster 10 - 'fire'.
As another way of assessing the quality of clustering we use documents - cluster representatives (document which is closest to the cluster center) and non-representatives (document which is farthest to the cluster center). So for example for Cluster 9 (not so obvious case) a representative document is:

> Vasario 3 d. 17 val. 10 min. į Vilniaus ligoninę kreipėsi ir dėl galvos smegenų sukrėtimo paguldytas R. I. (gimęs 1998 m.), kuris paaiškino, kad tą pačią dieną apie 9 val. Vilniuje, Rygos g., Jono Pauliaus II-ojo pagrindinėje mokykloje, jį sumušė 7-os klasės mokinys.[3]

While for Cluster 1 non-representative document is:

> <strong>Sveikatos sutrikdymai[4]

3 At 5.10 PM of February 3rd R.I. (born 1998) applied to Vilnius hospital and due to cerebral concussion was hospitalized. He explained that on the same day at 9 AM in a John Paul II school at Ryga street, Vilnius he was beaten by 7th grade student.

4 <strong> Health disturbances

which is a nice example of finding erroneous documents.

5.3. Classification: supervised grouping

Our last example of NLP for Lithuanian language is a document classification. It is a supervised machine-learning problem where, provided document features we try to predict its class. Originally Police reports are not classified, so as a first step we group documents performing simple keyword search on documents. We chose keywords: “eismas” (en: traffic), “gaisras” (en: fire), “grobimas” (en: robbery), “kyšis” (en: bribe), “lavonas” (en: corpse), “narkotikai” (en: drugs), “sužalojimas” (en: injury) and using them classify 7285 documents out of 14305. Of course, such attempt to classify documents might be (and actually is erroneous): one document might have several keywords. E. g., as a result of fire there might be corpses. Indeed in our case we have labeled 1038 documents as “fire”, but performing “corpse” search within these documents retrieves 415 documents.
Next we will use Complementary Naive Bayes classifier of Apache Mahout[5] to predict document class given its features (tf-idf weights of *n*-grams of up to 3 tokens).
Following table shows confusion matrix of classification:

Table 4. Classification confusion matrix.

lavonas (en: corpse)	**gaisras (en: fire)**	**kyšis (en: bribe)**	**grobimas (en: robbery)**	**narkotikai (en: drugs)**	**sužalojimas (en: injury)**	**eismas (en: traffic)**	**Classified as**	**Total**
1058	149	0	3	0	1	9	**lavonas (en: corpse)**	1220
265	741	0	2	0	0	30	**gaisras (en: fire)**	1038
0	0	731	0	2	0	131	**kyšis (en: bribe)**	864
1	1	0	1567	1	3	0	**grobimas (en: robbery)**	1573
2	3	3	5	235	2	1	**narkotikai (en: drugs)**	251
0	0	0	53	0	763	1	**sužalojimas (en: injury)**	817
14	2	38	11	1	0	1456	**eismas (en: traffic)**	1522

A detailed inspection of results shows that Bayes classifier reveals some new information. It shows that 265 documents assigned “fire” have something to do with “corpse” or that 149 documents assigned “corpse” have something to do with “fire”. Cross-misclassification of “bribe” and “traffic” is another interesting case which reveals and is confirmed by the detailed inspection of documents (and it is no surprise) that often bribe situation arises when traffic rules are violated. Consequently, we review our initial classification and again demonstrate that simple rule-based methods in combination with more sophisticated machine learning method lead to the more meaningful results and new insights, i. e. provides means not just to retrieve information, but to see relations within it, which is the desired aim of text analytics.

6. Demo search engine

Demonstrational version of the resulting system is available at http://policija.tokenmill.lt/. The system provides faceted search interface where facets, by means of techniques described above, are extracted from the texts of police event summaries. The facets allow filtering search results by persons or vehicles involved or tools used for committing offense.

7. Conclusions and further work

Our study results show that by combining Information Retrieval and Natural Language Processing techniques we can achieve better information recall and retrieval, as well, as better customer satisfaction. Moreover, we demonstrate a

5 https://cwiki.apache.org/MAHOUT/bayesian.html

need for further improvement and development of NLP techniques for Lithuanian language, because, even prototypes of stemmer (Krilavičius, Medelis 2010), Soundex (Krilavičius, Kuliešienė 2010) and improved linguistic rules as well as adapted ML techniques for Named Entities Recognition for Lithuanian language allow to achieve better results.
Future research plans include application of other machine learning (ML) and artificial intelligence (AI) techniques for named entities recognition, further improve tools for Lithuanian language analysis (e. g. stemmer, Soundex) and application of phonetic indexing techniques (e. g., Soundex, *n*-grams). Moreover, we are planning to investigate cross-language Information Retrieval, where, as a first step, we perform search on the press releases of the Ministry of Internal Affairs of the Republic of Azerbaijan, see http://infolex.tokenmill.lt/.

References

1. Baeza-Yates, R.; Ribeiro-Neto, B. 1999. Modern Information Retrieval. Addison Wesley.
2. Gate Development Team. 2010. Gate. http://gate.ac.uk/.
3. Holmes, D.; McCabe, M.C. 2002. Improving Precision and Recall for Soundex Retrieval. Proc. of International Conference on Information Technology: Coding and Computing: 22–27.
4. Lafferty, J.; McCallum, A.; Pereira, F. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. Proc. of the 18 Int. Conf. on Machine Learning: 282–289.
5. Kapočiūtė, J.; Raškinis, G. 2005. Rule-based annotation of Lithuanian text corpora. Information technology and control. Kaunas, Technologija, 34(3): 290–296.
6. Krilavičius, T.; Kuliešienė, D. 2010. Soundex for Lithuanian Language. Tech. rep., TokenMill.
7. Krilavičius, T.; Medelis, Ž. 2010. Porter Stemmer for Lithuanian Language. Tech. rep., TokenMill.
8. Lait A.J.; Randell B. 1996. An Assessment of Name Matching Algorithms. Tech. rep., No. 550. University of Newcastler upon Tyne.
9. Mahout Development Team. 2010. Mahout. http://mahout.apache.org/.
10. Manning, Ch.; Raghavan, P.; Schutze, H. 2008. Introduction to Information Retrieval. Cambridge Univ. Press.
11. Miller, G.A. 2009. WordNet - About Us. http://wordnet.princeton.edu.
12. Nadeau, D.; Sekine S. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes* 30(1): 3–26.
13. Porter, M.F. 1980. An algorithm for suffix stripping. Program, 14(3), 130−137.

Virginijus Kaušas. Experienced enterprise performance management consultant with 10 year experience in financial and performance analysis of corporations. Currently consulting large Lithuanian companies on business activities improvement investment projects.

Tomas Krilavičius. Formal methods and information retrieval expert. Over 15 years of experience in information and communication technologies. Currently associate professor in Vytautas Magnus University and senior researcher in Baltic Institute of Advanced Technologies. Defended his PhD thesis "Hybrid Techniques for Hybrid Systems" in Twente University (The Netherlands), in 2006.

Žygimantas Medelis. Information systems architect specializing in information retrieval and search solutions. Over 10 years experience in software design and implementation.

Danas Zuokas. Data scientist, PhD in Mathematics. More than 10 years of experience in mastering and applying various statistical methods. Currently is a lector at Vilnius University.