Jaume Baixeries, Dmitry I. Ignatov, Dmitry Ilvovsky, Alexander Panchenko (Eds.)

CDUD 2016 – The 3rd International Workshop on Concept Discovery in Unstructured Data

Workshop co-located with the 13th International Conference on Concept Lattices and Their Applications (CLA 2016)

July 18, 2016, Moscow, Russia

Volume Editors

Jaume Baixeries
Universitat Politècnica de Catalunya, Barcelona, Catalonia, Spain

Dmitry I. Ignatov,
Department of Data Analysis and AI, Faculty of Computer Science
National Research University Higher School of Economics, Moscow, Russia

Dmitry Ilvovsky,
Department of Data Analysis and AI, Faculty of Computer Science
National Research University Higher School of Economics, Moscow, Russia

Alexander Panchenko,
Language Technology, Computer Science Department
Technische Universität Darmstadt, Germany

Printed by the National Research University Higher School of Economics.

The proceedings are also published online on the CEUR-Workshop web site, Vol. 1625, in a series with ISSN 1613-0073.

Preface

Concept discovery is a subdomain of Knowledge Discovery (KDD) that uses human-centered techniques such as Formal Concept Analysis (FCA), Topic Modeling, Visual Text Representations, Conceptual Graphs etc. for gaining insight into the underlying conceptual structure of the data. Traditional machine learning techniques are mainly focusing on structured data whereas most data available resides in unstructured, often textual, form. Compared to traditional data mining techniques, human-centered instruments actively engage the domain expert in the discovery process.

This volume contains the papers presented at the 3rd International Workshop on Concept Discovery in Unstructured Data (CDUD 2016) held on July 18, 2018 at the National Research University Higher School of Economics, Moscow, Russia. This workshop welcomes papers describing innovative research on data discovery in complex data. It particular, it provides a forum for researchers and developers of text mining instruments, whose research is related to the analysis of linguistic and text data.

This year 15 papers had been submitted. Each submission has been reviewed, at least, by 2 program committee members. Seven papers have been accepted for regular publication in the proceedings, and three more submissions for publication as project proposals or abstracts.

Papers included in this volume cover a wide range of topics related to text mining and structures for text representation: text navigation, statistical learning models, automatic author or field identification in texts, among others.

An invited talk given by Natalia Loukachevitch from Moscow State University has opened the workshop program. She has surveyed modern tasks and approaches in sentiment analysis of Twitter messages.

Our deep gratitude goes to all the authors of submitted papers, as well as to the Program Committee members for their commitment. We also would like to thank our invited speaker and our sponsors: National Research University Higher School of Economics (Moscow, Russia), Russian Foundation for Basic Research, and ExactPro. Finally, we would like to acknowledge the EasyChair system which helped us to manage the reviewing process.

July 18, 2016
Moscow

Jaume Baixeries
Dmitry I. Ignatov
Dmitry Ilvovsky
Alexander Panchenko

Organisation

Program Chairs

Jaume Baixeries	Universitat Politècnica de Catalunya, Barcelona, Catalonia, Spain
Dmitry I. Ignatov	National Research University Higher School of Economics, Moscow, Russia
Dmitry Ilvovsky	National Research University Higher School of Economics, Moscow, Russia
Alexander Panchenko	Technische Universität Darmstadt, Germany

Program Committee

Simon Andrews	Sheffield Hallam University, UK
Aleksey Buzmakov	National Research University Higher School of Economics, Perm, Russia
Inma P. Cabrera	University of Malaga, Spain
Víctor Codocedo	LIRIS – INSA de Lyon, France
Pablo Cordero	Universidad de Málaga, Spain
Florent Domenach	Akita International University, Japan
Gillian Greene	Stellenbosch University, South Africa
Mehdi Kaytoue	LIRIS - INSA de Lyon, France
Jan Konecny	Palacky University, Olomouc, Czech Republic
Francesco Kriegel	Technische Universität Dresden, Germany
Sergei O. Kuznetsov	National Research University Higher School of Economics, Moscow, Russia
Victor Lempitsky	Skolkovo Institute of Science and Technology (Skoltech), Russia
Natalia Loukachevitch	Research Computing Center of Moscow State University, Moscow, Russia
Jesús Medina Moreno	University of Cádiz, Spain
Dmitry Mouromtsev	Nataional Research University ITMO, Saint Petersburg, Russia
Xenia Naidenova	Military Medical Academy, Saint Petersburg, Russia
Amedeo Napoli	INRIA – LORIA, Nancy, France
Alexey Neznanov	National Research University Higher School of Economics, Moscow, Russia
Jan Outrata	Dept. Computer Science, Palacky University, Czech Republic

Uta Priss	Ostfalia University of Applied Sciences, Germany
Artem Revenko	Semantic Web Company, Vienna, Austria
Barış Sertkaya	Frankfurt University of Applied Sciences, Germany
Dominik Slezak	University of Warsaw & Infobright, Poland
Rustam Tagiew	Polarez Engineering, Dresden, Germany
Martin Trnecka	Palacky University, Olomouc, Czech Republic
Dmitry Ustalov	N.N. Krasovskii Institute of Mathematics and Mechanics, Ural Branch of the RAS & Ural Federal University, Yekaterinburg, Russia
Sergey Zykov	National Research University Higher School of Economics, Moscow, Russia

Additional Reviewers

Ramírez Poussa, Eloisa

Sponsoring Institutions

National Research University Higher School of Economics, Moscow
ExactPro
Russian Foundation for Basic Research

Table of Contents

Invited Talk

Regular Papers

Research Proposals and Abstracts

Sentiment Analysis of Twitter Messages: Tasks, Approaches and Results

Natalia Loukachevitch

Research Computing Center, Moscow State University, Russia

Abstract. Microblog messages became a very popular tool for communication between people. Authors of the messages write about their life, convey their opinions on various topics including political and religious views, products and services, etc. Thus, microblogging sites as Twitter become valuable sources of information about peoples's opinions and sentiments. Approaches for extracting these opinions and their aggregation are actively studied.
In my talk I consider sentiment analysis tasks proposed for processing Twitter messages and the existing approaches including neural networks, which allowed improving the existing results during last year. Also I present results of the Russian evaluation of sentiment analysis systems (SentiRuEval) organized in 2015-2016.

Keywords: sentiments analysis, microblog messages, opinion mining

References

1. Loukachevitch, N.V., Rubtsova, Y.: Entity-oriented sentiment analysis of tweets: Results and problems. In: Text, Speech, and Dialogue - 18th International Conference, TSD 2015, Pilsen,Czech Republic, September 14-17, 2015, Proceedings. (2015) 551–559
2. Loukachevitch, N., Blinov, P., Kotelnikov, E., Rubtsova, Y., Ivanov, V., V, V., Tutubalina, E.: Sentirueval: testing object-oriented sentiment analysis systems in russian. In: Proceedings of International Conference Dialog. Volume 2. (2015) 3–13

Personalized Language Models for Computer-mediated Communication

Umme Hafsa Billah[1], Sheikh Muhammad Sarwar[2], Abdullah-Al-Mamun[3]

[1] Department of Computer Science and Engineering, University of Dhaka, Dhaka, Bangladesh
hafsabillah@yahoo.com
[2] Institute of Information Technology, University of Dhaka, Dhaka, Bangladesh
smsarwar@du.ac.bd
[3] Department of Computer Science and Engineering, University of Asia Pacific, Dhaka, Bangladesh
mamun05@uap-bd.edu

Abstract. In this paper, we investigate the performance of statistical language models on Instant Messaging (IM) data. Language Models (LM) are quite useful for modeling text data, and hence they are helpful in different contexts like *spelling correction*, *speech recognition*, *part-of-speech tagging* etc. Construction of LM on a users past messaging data would be a strategy to model her writing style, and that LM can then be used to predict the next word in her future communications. However, we hypothesize that a user follows a specific pattern of communication with each of her virtual acquaintances. As a consequence, LM built on her entire messaging history would degrade the performance of the next word predictor, while communicating with a specific person. In this paper, we deploy a special method that excludes some specific message contents from the entire history in order to build LM. Our method suggests that, at the time of communicating with a specific user, a special LM should be invoked from a set of models for increasing accuracy. We analyze the IM data of a set of users, and show that our method performs well in terms of perplexity.

Keywords: Language Model (LM), Perplexity

1 Introduction

People have conversation with each other almost every day using computing systems as a media; the applications or software used for this purpose are usually referred to as Instant Messaging (IM) system. It has become one of the mostly used paradigms for communication. As a result, it is integrated as a service with different types of social networks and e-mailing systems. As for example, Facebook and Gmail provide instant messaging facility as a part of their core services. People who use these systems, often need to communicate with friends, relatives, business collaborators etc. Generally, people use a certain way of typing, while

having casual conversation with another person. In order to facilitate and expedite such personalized typing, most IM software includes a component, that predicts and suggests a set of words, given the current input words of the message sender. Thus the next word prediction component is beneficial for everyday, as it reduces the time consumed for typing.

Human communication is predominantly personalized in real world *i.e.* a person internally uses and manages a specific dictionary for finding appropriate words to chat with another specific person. As for example, people exchange messages with their work groups formally using phrases like *With reference to our conversation previously, Yours sincerely* etc. On the other hand, at the time of exchanging messages with family or friends, they use casual phrases like *Hey wassup!, How you doing?* etc. Languages like Bengali are exposed to more personalized level of communication. For example, there are three different words for addressing a person: *tumi*, *tui* and *apni*, in Bengali against a single word *you* in English. Based on these three types of addressing, for a single sentence in English, there can be three possible sentences in Bengali with the same meaning. Some samples are shown in table 1. Thus, a personalized prediction system would be a contribution to gear up the typing speed while having informal computer based communication; especially for the case of languages like Bengali. As a result, the need for constructing personalized language based statistical models can never be obviated and undervalued.

Table 1. Sentence Variations in English and Bengali

English	Bengali
How are you?	Apni kemon achen? (Formal Style)
	Tumi kemon acho? (Semi-Formal Style)
	Tui kemon achish? (Informal Style)

Researchers have taken keen interest on building personalized language models for different purposes. In 2009, Xue et al. proposed a method for personalizing search results based on user interest [1]. They modeled individual profile using statistical language models, and finally constructed clusters to form group models. The models incorporated in a cluster were gathered from people who have same taste in web content. Li et al. used statistical language models for personalizing information extraction services *i.e.* text snippet extraction [2].

The existing methodologies for personalized word prediction emphasizes mostly on estimation based upon the built-in statistical language models, which are consisted of using the dictionary of a particular language. However, for phonetic typing the task is not that simple, as the spelling of a word may differ from user to user and deviate from the standard form, if one is considered as standard. Apart from this, several other issues compound the task, and we look forward to develop a personalized next word predictor as a remedy to this situation. Based on the problems mentioned above, we would like to address the following research questions in this work :

- Will personalized language models improve the next word prediction component used by an IM?
- How the sources of data for training language models effect personalization?
- How can we measure the performance improvement of such systems by creating a proper data set and evaluation strategy?

In order to answer these questions, we would like to develop person specific statistical language models, of which one will be invoked, when a person is communicating with another person. Till this end, we have come up with two hypotheses:

- A single user follows a specific linguistic style while communicating with another person
- Excluding data that degrades a language model, can improve the performance of the model in the context of improving the next-word prediction component of an IM service

Based on these two assumption we modeled specific persons style by building statistical language models with an exclusion method. Thus, the key contributions of this work are :

- Building language models following a users linguistic style especially in Bengali. The linguistic style is captured based on the interaction of a user with other users.
- An exclusion method based language model which would exclude the unnecessary information from the model and would produce better suggestions for user.

2 Related Works

In this section, we review the background literature related to our personalized next word prediction strategy. A generalized word prediction system was proposed by Bosch, which could predict millions of words per second [3]. He used a simple decision-tree algorithm that was less costly in terms of complexity, in order to use a large amount of data for training from the *Reuters* corpus. However, a personalization component was not included in this method, as it was not developed considering the dynamics of human communication. Siska et al. designed an adaptive keyboard that could adjust its predictive features and key displays based on current user input [4]. They implemented the personalized word prediction module using common English dictionary to improve the performance of such a system. The built-in English dictionary was used with an existing database that the system needed to overwrite personalized phonetic words. Nonetheless, this method requires a huge database of training corpora which is not suitable for a smart-phone based implementation.

A learning approach employing hierarchical modeling of phrases was proposed by Richard et al. [5]. This approach reduced the amount of initial training data required to facilitate on-line personalization of the text prediction system.

It is also intended for the development of assistive technologies for disabilities, especially within the domain of augmentative and alternative communications (AAC) devices. The key insight of the proposed approach is the separation of stop words, which primarily play syntactical roles in phrases. Matthew suggested a system to improve the rate at which users can participate in a conversation using an AAC (Augmentative and Alternative Communication) device. This was intended for persons who are unable to communicate verbally [6].

Author profiling techniques were also used for personalizing messaging systems, and most of these systems are based on machine learning approaches. Tayfun et al. proposed to investigate the possibility of predicting several users and message attributes in text-based, real-time, on-line messaging services [7]. Specifically, they aimed to identify instant message authors correctly using style-based approach. Inches et al. designed a framework for identifying topic and author from on-line user-generated conversations [8]. They used different similarity metrics to identify document features and took an entropy-based approach to identify authors. Author identification have been improvised a step further by Villatoro-Tello et al. where they identified misbehaving authors in instant messaging by classifying user text and building models based on SVM and neural networks [9].

Sarwar et al. showed that constructing a LM with the conversation text pair of users, and trying to predict the text of other users provides different outcomes for different users. Even though it seems quite intuitive, the outcome of this research indicated that a LM built on a conversation text could be useful to predict the text of a cluster of users [10].

3 Background

In this section we explain two necessary topics that are essential to our proposed method: *language model* and *perplexity*.

3.1 Language Model

Language models (LM) are heavily used in many applications using Machine Translation and Speech Recognition technology. Language models are used to evaluate the probability of a sequence of words. Given a sequence of words of length m, it is possible to estimate the probability of the sequence $P(w_1, w_2, ..., w_m)$, using LM [11]. Based on the context there are different types of LM. If the probability of a word w_k, depends on its previous word w_{k-1}, then it is denoted as bi-gram LM. However, in general LM are defined as n-gram language model, where the probability $P(w_1, \ldots, w_m)$ of observing the sentence $w_1, \ldots, w_m$ is approximated as shown in Equation 1.

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1}) \tag{1}$$

The joint probability distribution can be estimated as below:

$$\prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1}) = \frac{count(w_1, \ldots, w_{i-1}, w_i)}{count(w_1, \ldots, w_{i-1})} \tag{2}$$

In case of bigram language model Equation 2, can be re-written as Equation , based on markov assumption.

$$\prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1}) = P(w_1) \prod_{i=2}^{m} P(w_i \mid w_{i-1}) \tag{3}$$

In these paper we have used bigram language model to extract the linguistic style of an author.

3.2 Perplexity

Perplexity is a measure that is used to test the quality of LM. In order to test LM, test data is used and perplexity is measured. Let us assume that there are m sentences in test data: $t_1, t_2, \ldots, t_m$. It is possible to measure the log probability of each sentences using LM:

$$\log \prod_{i=1}^{m} P(t_i) = \sum_{i=1}^{m} logP(t_i) \tag{4}$$

Now, Perplexity (PP) can be defined using the following equation:

$$PP = 2^{-l}, \; where \quad l = \frac{1}{M} \sum_{i=1}^{m} logP(t_i) \tag{5}$$

in 5, M is the total number of words in the test data. The lower the value of perplexity the better the LM are. The worst possible LM results in the number of words in the test data. Perplexity is a measure of effective *branching factor* [12].

4 Proposed Method

The proposed method is developed based on the hypothesis that *A single user follows a specific linguistic style while communicating with another person.* Thus, at first, our aim is to construct a collection of personalized dictionaries *i.e.* language models for a user. Finally, those models would be used to predict the next word for the user at the time of sending instant messages. The invocation of a specific model would be completely dependent on the person, with whom the user will be communicating.

Language models can assign a probability value to a word given a sequence of words. For example, using *bigram* language model, we can predict the probability of a word "computer" given the word "personal". Moreover, using language model notation, we can represent it as $P(computer \mid personal)$. Using

this value, we want to estimate the probability of the word "computer", given the word "personal". To describe our method, we use the terms *language models* and *models*, interchangeably. In the following paragraphs, we would discuss our methodology from the perspective of a single user (u), for whom we would build a set of models (M), which will facilitate his computer-mediated communication with other users (U) in his network. Moreover, there would be a one-to-one relationship between M and U, *i.e.* $|M| = |U|$.
In order to describe the method, we consider that user u has connection with a set of k users $U = \{u_1, u_2, u_3, \ldots, u_k\}$, through an instant messaging service. Interaction set $I = \{i(u, u_1), i(u, u_2), \ldots, i(u, u_k)\}$ contains all the messages sent to each $u_k \in U$ by user u. Hence, we are only considering the unidirectional messages sent by user u to all other users. According to our own definition these messages form a General Dictionary (GD_u), which we use to build a generalized model for u.

According to the first part of our research hypothesis, GD_u can not be a suitable source of observed data to build a generative model, which can be used to predict the chat content of u and u_k. As the instant messaging content of a user varies significantly, based on the other person he is communicating with, GD_u would be a source of data that would degrade the model. Some conversations in GD_u can lead to the development of inefficient models, and building a next word predictor based on those models would not improve the communication speed. Thus, in order to model the interaction $i(u, u_k)$, we would need a distinct model $m(u, u_k)$, and it should be built on $I_k \subset I$. This would result in a model set $M = \{m(u, u_1), m(u, u_2), \ldots, m(u, u_k)\}$. Now, when u would be communicating with u_k, $m(u, u_k)$ would be invoked to generate words for u.

The second part of our hypothesis is about the construction of I_k. If we exclude a subset of interactions $\bar{I}$ from I, we would be able to get textual contents that model the conversation between u and u_k more closely. Thus, we can construct I_k using the following equation:

$$I_k = I - \bar{I} \tag{6}$$

From equation 6, it can be seen that $\bar{I}$ is a cluster of interactions, which we will exclude from I. Our goal is to construct $m(u, u_k)$ using I_k. In order to build $m(u, u_k)$, we construct one model for each interaction from $I - i(u, u_k)$. After that we evaluate the perplexity of each model on the held out data from interaction $i(u, u_k)$. After that we select top-n models that result in highest perplexity values, and create interaction set $\bar{I}$, by including the associated interactions with them. We also refer $\bar{I}$ as the Worst Interaction Set (WIS) for the ease of understanding. Thus, we are trying to estimate, which models maximize the uncertainty, while predicting the held out data. We hypothesize that the associated interactions used to build these models introduce more uncertainty in GD_u. By excluding $\bar{I}$ from I, and constructing a model on I_k, we reduce the entropy in GD_u. As a consequence, the final model $m(u, u_k)$ would be a better predictor than a generalized model constructed from GD_u.

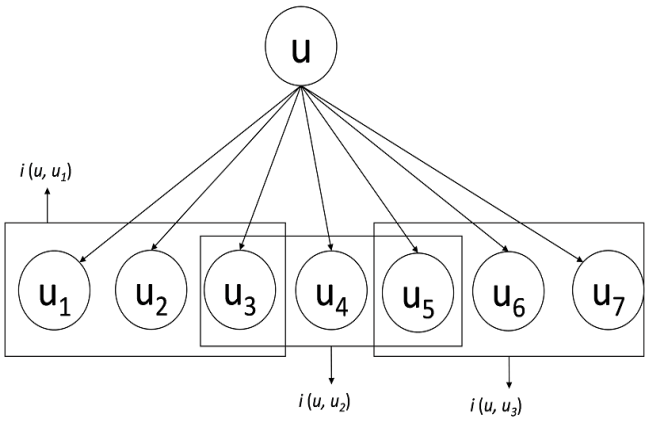


Fig. 1. Interaction clustering based on top-3 interactions

We have shown a sample execution of our proposed method using Figure 1. Initially, we create a LM based on $i(u, u_1)$ and evaluate the perplexity of the model using the Equation 5, for all the users $u_j \in U$. As a result, for each $u_j \in U$, we get a perplexity value. The worst perplexity of a LM on a test data is the number of words in the test data. According to the scope of our work, we only consider bigram based LMs.

After obtaining the perplexity values for each $i(u, u_j) \in I$, we sort them in descending order and select the top-n interactions. We extract the user id $u_t \in U$ from the interactions and create a user group with those values. We perform this process repeatedly by building LM with all the interactions from I one by one. From Figure 1, it can be observed that for interaction $i(u, u_4)$, three interactions have been grouped together: $i(u, u_1)$, $i(u, u_2)$ and $i(u, u_3)$. As these three interactions produced three highest values of perplexity with the LM constructed using $i(u, u_4)$, they are grouped together.

5 Experimental Setup

5.1 Data Set

We have collected the summary and analysis using our program from the chat logs of three different facebook users. Chatting data is completely private and we did not collect the data from users, instead we provided our program to the users and they gave us the output generated from the program. All the users were IT professionals; they could run our program to generate summary data for us. Each of the users, who ran our program, communicated with at least 7 different people and their basic interaction was in Bengali; the total collection

contained 22 interactions. Prior to running our program, a privacy agreement was signed by each user.

Testing and training data set was created from the chat logs by our script. Last 20% data of each interaction of a user was kept as test data. We trained our model on the first 80% data and evaluated the model on the last 20% held out data. In table 2, some properties of our data set are shown formally.

Table 2. User chat log data properties.

User	Average sentence length per line	Total words per interaction	No of Interactions
101	14	5466	7
102	18	3555	8
103	24	5782	7

It can be seen from Table 2 that we have given each user a unique identifier, so that he or she can be remained anonymous. According to the table, User 101 has 14 sentences on an average in each interaction set, with a total of 5466 words. It is also evident that user 101 interacted with 7 persons in total. Each individual user was asked to provide his messaging content considering different groups of people like family, friends, cousins, colleagues etc. so that we can get different types of interactions.

5.2 Experimental Setup and Result

In this work, we have tried to select the best model that performs well in terms of perplexity, on the held out data of a specific user interaction. At first, we create a generalized bigram model over all the user interactions I. In this paper, we use the term General Dictionary (GD) in exchange with I for the ease of understanding. After the general bigram model is created, we use it to evaluate the performance of GD for all the interactions of a specific user. Then we create a specialized LM, namely WIM for each interaction by subtracting $\bar{I}$ from I. For the experimentations in this paper, we subtract top-3 interactions from GD, and build models on resultant data. Finally, each of the models is evaluated based on the calculation of perplexity on the held out data of each user interaction. The percentage improvement of WIM with respect to GD is calculated and shown in all our result tables. A negative value depicts poor performance of WIM, whereas a positive value represents performance improvement. The complied results from the experimentation for each user are shown using Table 3, Table 4 and Table 5, respectively.

From Table 3, we can see that user 101 interacts with a total of 7 persons with 7 different ID's. For interaction (101,201) its worst interaction set WIS consists of the user with ID's 202, 203, 206. This means that these interactions actually

degrade the performance of the language model built for user 101, using the GD. Here, the WIM improves the model by 9.49% which is considerably higher than GD. However, in interaction (101,206) we can see that WIM actually gives 18.6% poor result comparing to GD. It is observed that those cases are very rare, when GD outperforms WIM.

Table 3. Different LM result on user 101 chat log.

Interaction (u_i,u_j)	WIM	WIM Perplexity	GD Perplexity	Improvement of WIM over GD(%)
(101,201)	{(101,202),(101,203),(101,206)}	14.27	15.76	9.49
(101,202)	{(101,205),(101,207),(101,206)}	16.89	19.280	12.36
(101,203)	{(101,202),(101,205),(101,206)}	18.81	21.36	11.95
(101,204)	{(101,201),(101,202),(101,206)}	17.83	19.39	8.00
(101,205)	{(101,202),(101,207),(101,206)}	19.72	21.47	8.17
(101,206)	{(101,203),(101,205),(101,204)}	31.77	26.78	-18.66
(101,207)	{(101,205),(101,202),(101,206)}	15.21	17.22	11.66

Table 4. Different LM result on user 102 chat log.

Interaction (u_i,u_j)	WIM	WIM Perplexity	GD Perplexity	Improvement of WIM over GD(%)
(102,301)	{(102,308),(102,306),(102,304)}	30.04	12.35	-143.28
(102,302)	{(102,308),(102,306),(102,304)}	21.37	23.79	10.18
(102,303)	{(102,306),(102,307),(102,30)8}	11.47	12.82	10.51
(102,304)	{(102,305),(102,308),(102,306)}	12.46	13.59	8.36
(102,305)	{(102,301),(102,308),(102,304)}	10.49	12.01	12.68
(102,306)	{(102,305),(102,301),(102,304)}	12.61	14.32	11.90
(102,307)	{(102,301),(102,306),(102,304)}	12.19	14.70	17.10
(102,308)	{(102,305),(102,304),(102,307)}	11.74	13.85	15.26

In table 4, the interaction between user 102 and other users are shown. Here, we can see that while interacting with user 301, WIM gives worse result comparing to GD. In all the other cases, WIM performs significantly better than GD.

Table 5 shows the performance on the interactions of user 103. In the interactions (103,402), (103,403), (103,404) WIM performs poorly giving the improvement percentage -13.57%, -79.64%,-13.63% respectively. However, in the interaction (103,403) the result is very poor in comparison with GD.

Table 5. Different LM result on user 103 chat log.

Interaction (u_i,u_j)	WIM	WIM Perplexity	GD Perplexity	Improvement of WIM over GD(%)
(103,401)	{(103,406),(103,404),(103,402)}	10.16	11.83	14.10
(103,402)	{(103,405),(103,404),(103,403)}	12.34	10.87	-13.57
(103,403)	{(103,406),(103,407),(103,405)}	18.11	10.08	-79.64
(103,404)	{(103,405),(103,402),(103,406)}	13.31	11.71	-13.63
(103,405)	{(103,406),(103,401),(103,404)}	10.55	11.87	11.14
(103,406)	{(103,403),(103,404),(103,405)}	8.98	9.86	8.93
(103,407)	{(103,404),(103,402),(103,405)}	8.49	10.32	17.73

In our experiment, we have shown that the language models built by excluding the Worst Interaction Set (WIS) from I improves the performance of the general dictionary based LM. By excluding WIS, we actually remove the contents, which affect the performance. However in some cases, we have found that excluding WIS from I doesn't always improve the performance; in fact in some situations GD outperforms WIM.This phenomenon occurs, because we have subtracted a fixed number of interactions from GD for our experiment. Moreover, there are some interactions in WIS cluster, which might generate important suggestions for user. By excluding them, we are removing those important information from GD, which results in poor perplexity scores. As a result, it can be experimentally inferred that excluding WIS from the interaction set will build better LM than the LM built over the generalize dictionary for a single user. But, in this paper, we have conducted small experimentation, and publish the results after running our program with input from three users only. Therefore, even though the results are quite interesting, we can not finally conclude that excluding information from the GD of a user will model her conversation more accurately.

6 Conclusion

The research leads to the development of a user-oriented and personalized next-word predictor for instant messaging, which can speed up the text-based communication among different people in the virtual world. The ever-growing field of social media and instant messaging have created the necessity to design a system that could support fast, comfortable and smooth typing. Even though we have shown our result in terms of a standard NLP metric, perplexity, we hope to implement an instant messaging system for the on-line evaluation of our idea. Moreover, we would like to collect more user chat log with privacy agreement, anonymize our data set using some well known anonymization algorithms like k-anonymization and publish our data set in future. Besides, we would try to filter out some unnecessary information *i.e* emoticon, stop words, punctuation marks etc. which will improve the performance of the language models.

Acknowledgment This work is supported by the University Grant Commission, Bangladesh under the Dhaka University Teachers Research Grant No-Regi/Admn-3/2016/46897.

References

1. Gui-Rong Xue, Jie Han, Yong Yu, and Qiang Yang. User language model for collaborative personalized search. *ACM Transactions on Information Systems (TOIS)*, 27(2):11, 2009.
2. Qing Li and Yuanzhu Peter Chen. Personalized text snippet extraction using statistical language models. *Pattern Recognition*, 43(1):378–386, 2010.
3. Antal Van Den Bosch. Scalable classification-based word prediction and confusible correction. *Traitement Automatique des Langues*, 46(2):39–63, 2006.
4. Siska Fitrianie and Leon Rothkrantz. An adaptive keyboard with personalized language-based features. In Vaclav maousek and Pavel Mautner, editors, *Text and Speech and Dialogue 2007*, number LNAI. Springer, Springer, sep 2007.
5. Richard Gabriel Freedman, Jingyi Guo, William H. Turkett Jr., and Victor Pal Pauca. Hierarchical modeling to facilitate personalized word prediction for dialogue. In *AAAI Workshop: Plan, Activity, and Intent Recognition*, volume WS-13-13 of *AAAI Workshops*. AAAI, 2013.
6. Matthew E. J. Wood. Syntactic pre-processing in single-word prediction for disabled people. Technical report, Bristol, UK, UK, 1996.
7. Tayfun Kucukyilmaz, B. Barla Cambazoglu, Cevdet Aykanat, and Fazli Can. Chat mining: Predicting user and message attributes in computer-mediated communication. *Inf. Process. Manage.*, 44(4):1448–1466, July 2008.
8. Giacomo Inches and Fabio Crestani. Online conversation mining for author characterization and topic identification. In *Proceedings of the 4th workshop on Workshop for Ph. D. students in information & knowledge management*, pages 19–26. ACM, 2011.
9. Esaú Villatoro-Tello, Antonio Juárez-González, Hugo Jair Escalante, Manuel Montes-y Gómez, and Luis Villasenor Pineda. A two-step approach for effective detection of misbehaving users in chats. In *CLEF (Online Working Notes/Labs/Workshop)*, 2012.
10. Sheikh Muhammad Sarwar and Abdullah-Al-Mamun. Next word prediction for phonetic typing by grouping language models. In *2016 2nd International Conference on Information Management (ICIM)*, pages 73–76. IEEE, 2016.
11. Jianfeng Gao, Joshua T. Goodman, and Jiangbo Miao. The use of clustering techniques for asian language modeling, 2001.
12. Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. Perplexity - a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63, 1977.

Framework for Conceptual Modeling on Natural Language Texts

Mikhail Bogatyrev, Kirill Samodurov

Tula State University, Tula, Russia

okkambo@mail.ru, zmeymc@gmail.com

Abstract. The paper presents the framework for conceptual modeling which has been used in on-going project of developing fact extraction technology on textual data. The modeling technique combines the usage of conceptual graphs and Formal Concept Analysis. Conceptual graphs serve as semantic models of text sentences and the data source for formal context of concept lattice. Several ways of creating formal contexts on a set of conceptual graphs have been investigated and resulting solution is proposed. It is based on the analysis of the use cases of semantic roles applied in conceptual graphs and their structural patterns. Concept lattice building on textual data is interpreted as storage of facts which can be extracted by using navigation in the lattice and interpretation its concepts and hierarchical links between them. Experimental investigation of the modeling technique was performed on the annotated textual corpus consisted of descriptions of biotopes of bacteria.

Keywords: conceptual modeling, conceptual graphs, concept lattice, biotopes of bacteria.

1 Introduction

Conceptual modeling in the Natural Language Processing (NLP) is a way of modeling semantics. Semantics of texts is transformed to semantics of conceptual models at a high level of abstraction, in terms of concepts. Conceptual graphs (CGs) [22] represent a well-known type of conceptual models and there are some applications of them in Text Mining problems solutions [13, 15].

Another paradigm of conceptual modeling is Formal Concept Analysis [10]. It is a mathematical theory of data analysis which studies how objects can be hierarchically grouped together according to their common attributes. Strong mathematical background of FCA (it is based on the lattice theory [2] and uses matrix model of so named “formal context”) provides its implementations as rigorous instrument for

Information Retrieval (IR). The number of FCA applications now is growing up including applications in Text Mining and linguistics [6, 19]. It is also applied in more general field of knowledge processing [17].

The idea of joining two paradigms of conceptual modeling - conceptual graphs and concept lattices - seems very attractive but not elaborated in the FCA community. There are several its realizations due the years, from early implementation in [23] up to recent investigations in [9].

This idea may get a second breath when FCA is utilized on textual data and conceptual graphs serve as conceptual model of text semantics. Acquiring conceptual graphs from natural language texts is non-trivial problem but it is quite solvable [5, 14]. The concepts of conceptual graphs may be treated as objects and attributes for formal context as far as the "attribute" conceptual relation really exists in conceptual graphs acquired from natural language texts. Actually, as it is followed from our investigations, the "attribute" relation is not always good and even enough for formal context. Except the "attribute" conceptual relation some other relations must be analyzed in conceptual graphs to find objects and attributes needed for formal context.

The main problem which arises in CGs – FCA applications is the problem of building formal concepts on conceptual graphs. Solution of this problem and the whole principle of applying FCA on textual data are closely depended on the real-life problems have been solved with FCA on textual data [12, 16]. In the sense of Information Retrieval these problems may be generalized to the fact extraction problem. Using FCA in its solution is based on that concept lattice built on textual data may be interpreted as storage of facts which can be extracted by using navigation in the lattice and interpretation its concepts and hierarchical links between them.

One of the fields where Text Mining applications are growing rapidly is Bioinformatics. New term of Biomedical Natural Language Processing (BioNLP) has been appeared there [1]. This is stipulated by huge amount of scientific publications in Bioinformatics and organizing them into corpora with access to the full texts of articles. FCA has great potential to take up a challenge from such areas as BioNLP.

In this paper we present the framework for conceptual modeling which has been used in on-going project of developing fact extraction technology on textual data.

The next section of the paper contains brief description of FCA basics and conceptual modeling technique which is used in the framework.

Section 3 is devoted to the framework; its structure and functionality are described there.

In the section 4 current experimental results of using framework on bacteria biotope textual corpus are presented and section 5 contains conclusion and planning future works.

2 CGs – FCA modeling on natural language texts

We are developing conceptual modeling technique which combines the usage of conceptual graphs and conceptual lattices from Formal Concept Analysis. Consider some FCA basics needed for understanding the modeling technique.

2.1 Formal Concept Analysis basics

There are two basic notions FCA deals with: *formal context* and *concept lattice*. Formal context is a triple $\mathbf{K} = (G, M, I)$, where G is a set of objects, M – set of their attributes, $I \subseteq G \times M$ – binary relation which represents facts of belonging attributes to objects. The sets G and M are partially ordered by relations ϕ and $-$, correspondingly: $G = (G, \phi)$, $M = (M, -)$. Formal context may be represented by [0, 1] - matrix $\mathbf{K} = \{k_{i,j}\}$ in which units mark correspondence between objects $g_i \in G$ and attributes $m_j \in M$. The concepts in the formal context have been determined by the following way. If for subsets of objects $A \subseteq G$ and attributes $B \subseteq M$ there are exist mappings (which may be functions also) $A' : A \to B$ and $B' : B \to A$ [1] with properties of $A' := \{\exists m \in M \mid < g, m > \in I \ \forall g \in A\}$ and $B' := \{\exists g \in G \mid < g, m > \in I \ \forall m \in B\}$ then the pair (A, B) that $A' = B$, $B' = A$ is named as formal concept. The sets A and B are closed by composition of mappings: $A'' = A$, $B'' = B$; A and B is called the *extent* and the *intent* of a formal context $\mathbf{K} = (G, M, I)$ respectively.

A formal concept is a pair (A, B) of subsets of objects and attributes which are connected so that every object in A has every attribute in B, for every object in G that is not in A, there is an attribute in B that the object does not have and for every attribute in M that is not in B, there is an object in A that does not have that attribute.

The partial orders established by relations ϕ and $-$ on the set G and M induce a partial order $\leq$ on the set of formal concepts. If for formal concepts (A_1, B_1) and (A_2, B_2), $A_1 \phi A_2$ and $B_2 - B_1$ then $(A_1, B_1) \leq (A_2, B_2)$ and formal concept (A_1, B_1) is less general than (A_2, B_2). This order is represented by *concept lattice*. A lattice consists of a partially ordered set in which every two elements have a unique *supremum* (also called a least upper bound or *join*) and a unique *infimum* (also called a greatest lower bound or *meet*).

According to the central theorem of FCA [10], a collection of all formal concepts in the context $\mathbf{K} = (G, M, I)$ with subconcept-superconcept ordering $\leq$ constitutes the *concept lattice* of $\mathbf{K}$. Its concepts are subsets of objects and attributes connected each other by mappings A', B' and ordered by a subconcept-superconcept relation.
Although that level of abstraction makes FCA suitable for use with data of any nature, its application to specific data often requires special investigation. It is fully relevant for using FCA with textual data.

---

[1] More rigorous definition assumes that these mappings are different: $\varphi : A \to B, \psi : B \to A$ but it is not a matter of principle here.

2.2 FCA on textual data

The main problem in applying FCA on textual data is the problem of building formal context. If textual data is represented as natural language texts then this problem becomes especially important.

There are several approaches to the construction of formal contexts on the textual data, presented as separate documents, as data corpora. One, mostly applied variant of context is that its objects are text documents and the attributes are the terms in these documents [6, 7]. The main problem which can be solved with that formal context and concept lattice is the problem of retrieving textual documents.

Another variant of formal context is building directly on the texts. In the general case, various word combinations constitute its concepts and the number of such concepts may be very large. An advantage of such variant is that this context contains potentially more information about texts than previous one and more general problems such as fact extraction problem can be solved on that formal context. The disadvantage of it is its great dimension and possible many pointless concepts.

Restricting the dimension of formal context and giving it more semantics is doing by representing in it the various features of its source texts: semantic relations (synonymy, hyponymy, hypernymy) in a set of words for semantic matching [12], verb-object dependencies from texts [7], words and their lexico-syntactic contexts [16].

For building formal context, one needs to distinguish some of these lexical elements in texts as objects and attributes. There are following approaches to solve this problem:

- adding special descriptions to texts which mark objects and attributes and partial order – this is usually done manually;
- using semantic models of texts and corpus tagging [7].

We apply the second approach and use conceptual graphs for representing semantics of individual sentences of a text.

2.3 CGs – FCA modeling process

The whole process of CGs – FCA modeling has the following steps.

1. *Acquiring a set of conceptual graphs from input texts*. Conceptual graph [22] is bipartite directed graph having two types of vertices: concepts and conceptual relations. These vertices are connected by arrows representing binary relations. Conceptual graphs can be created by our tool CGs Maker [2]. Some details about it can be found in [13, 14].

2. *Aggregating the set of conceptual graphs*. Aggregation is needed to exclude excessive dimension of conceptual models, not related to useful information. We have tested two ways of conceptual graphs aggregation: conceptual graphs clustering and restricting the number of conceptual graphs by identifying and excluding sentences which are not corresponded to the problem solving with the current technique.

[2] The lightweight online version of CGs Maker for simple English and Russian texts can be found at http://85.142.138.156:8888 .

3. *Creating formal contexts*. One or several formal contexts are built on the aggregated conceptual graphs. The number of formal concepts and the method of building them have been determined in the solving problem.

4. *Building concept lattice*. Having a concept lattice, it is possible to identify connections between the concepts according to the principle of "common – particular". Each concept, the node in the lattice is interpreted as the set of potential facts of certain level, which is associated with other facts.

5. *Fact extraction from concept lattice*. Concept lattice is the data storage for fact extraction system. This system has domain oriented user interface for query processing and generating output.

This paper reflects results of investigations corresponded to steps 1-3 of the process. On the step 4 we used standard open source tool for building and visualizing concept lattices [8] which we integrated into the whole modeling system. Creating the fact extraction system (step 5) is separate problem currently being under development.

2.4 Usage of conceptual graphs

The crucial step in the described process of CGs – FCA modeling is creating formal contexts on the set of conceptual graphs. At first glance, this problem has simple solution: those concepts which are connected by "attribute" relation have been put into formal context as its objects and attributes. Actually the solution is much more complex. To illustrate it consider conceptual graph for the sentence "*Xylella fastidiosa is a gram-negative fastidious, xylem-limited bacterium*" shown on Fig. 1. This sentence is from bacteria biotopes textual corpus [4] which we use for our method evaluation.

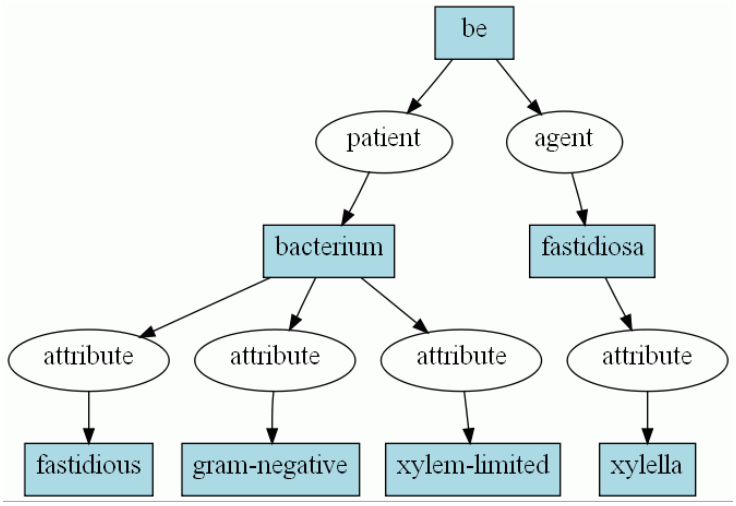


Fig. 1. Conceptual graph for the sentence "Xylella fastidiosa is a gram-negative fastidious, xylem-limited bacterium."

Conceptual graph on the Fig.1 has four conceptual relations “attribute” but only three of them indicate real objects and attributes for formal context. Using “fastidiosa” as object and “Xylella” as attribute in the formal context is wrong way because “Xylella fastidiosa” is known full name of this bacterium. Full names of bacteria have to be objects in the formal context devoted to bacteria. Word combinations denoting the names of bacteria must be recognized before conceptual graphs building. There is no other way of doing this than to use an external source of information, for example, the corpus tagging.

We also realize the following rules for creating formal contexts on conceptual graphs.

1. Not only individual concepts and relations, but also patterns of connections between concepts in conceptual graphs represented as subgraphs have been analyzed and processed. The pattern “agent - patient” is mostly frequent in biotope texts.
2. The hierarchy of conceptual relations in conceptual graphs is fixed and taken into account when creating formal context. Such hierarchy exists on the Fig.1: relations “agent” and “patient” are on the top level and relations "attribute" belong to underlying level. Using this hierarchy of conceptual relations we can select for formal contexts more or less details from conceptual graphs. This makes conceptual graphs more power and flexible semantic model for FCA than n-grams or collocations.
3. FCA – model for fact extraction is domain specific. Domain information is also taken into account in conceptual graphs building. This information is from external resources – thesauruses or tagging of textual corpuses.

   Concrete implementations of these rules are in the section 4.

3 Architecture and Functionality of the Framework

Architecture of the CGs – FCA modeling framework is shown on the Fig. 2. Consider its main elements.

Database. Database is very important part of the framework. We use relational database on the SAP-Sybase platform. It was built with CASE technology PowerDesigner™ [18] and may be scaled and expanded. Database stores texts, conceptual graphs, formal contexts and concept lattices. Special indexing is applied to textual data.

Conceptual graphs building module. This module and several other modules constitute the NLP block of modules of the framework. They realize our algorithm of acquiring conceptual graphs from texts, visualization of conceptual graphs and their clusters, interaction with external resources including WordNet.

English and Russian languages have been supported in the framework. The framework has internal dictionaries and may communicate with external ones.

Representing of modeling results. Modeling results have been presented as visualization of conceptual graphs and concept lattices as in table and textual forms. Storing all objects in database allows analyzing its data and computing conceptual graphs and concept lattice characteristics.

Programming environment. Java is the main programming platform which is used in the framework. Some modules of NLP block have been written on PowerScript language of SAP-Sybase platform.

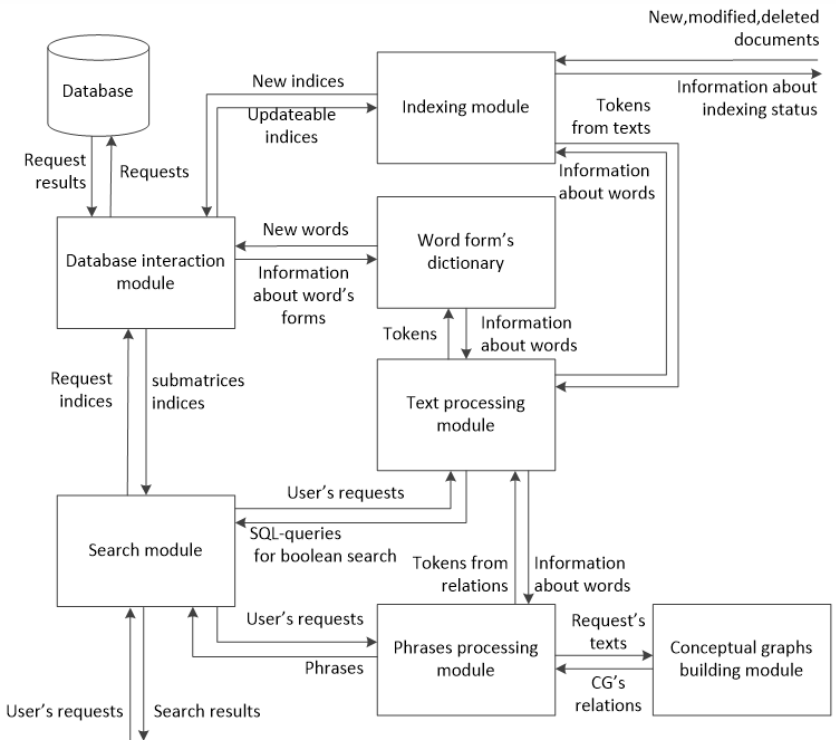


Fig. 2. Architecture of the framework

4 Experiments and Results

Experimental evaluation of CGs – FCA modeling technique has been carried out on the textual corpus of bacteria biotopes which is used in the innovation named as BioNLP Shared Task [4]. This innovation includes three IR tasks: the Bacteria Gene Renaming, the Bacteria Gene Interaction and the Bacteria Biotopes. The Bacteria Biotope task is formulated as consisting of two standard Text Mining tasks of Named Entity Recognition (NER) and Relations Extraction (RE) [20].

Biotope is an area of uniform environmental conditions providing a living place for plants, animals or any living organism. According to [4] there are two types of entities to be extracted: the names of bacteria and their locations. We added third entity of pathogenicity of bacteria.

It is preliminarily clear that the task of extracting the names of bacteria and the task of extracting locations and pathogenicity have different complexities. For extracting the names of bacteria some words or collocations (*Xylella fastidiosa*) have to be analyzed in the text. Locations and pathogenicity may be represented by more complex and long word combinations. As for bacterium *Xylella fastidiosa* on the Fig. 1, the example of its location is the following fragment from the text about it: "*the bacteria … receive a safe environment and metabolites from the insect*". To extract "*insect"* as location of bacterium we need to analyze some relations between words in the sentence. This is done also through the use of conceptual graphs.

Biotope texts tagging includes full names of bacteria, its abbreviated names and unified key codes in the database. We add additional tags if special words (*extreme, obligately*, etc.) recognized in the texts.

A BioNLP data is always domain-specific. All the texts in the corpus [4] are about bacteria themselves, their areal and pathogenicity. Not every text contains these three topics but if some of them are in the text then they are presented as separate text fragments. This simplifies text processing. According to these three topics of interest we construct three different formal contexts of "Entity", "Areal" and "Pathogenicity". They engender three different concept lattices which are connected each other. To join lattices we use facet technology [19].

Our solution of the task of Named Entity Recognition is supported by conceptual graphs. As it is illustrated above (Fig. 1) conceptual graphs can represent names of bacteria as named entities. Named Entity Recognition also includes anaphora resolution.

4.1 Anaphora resolution and noise reduction

Anaphora resolution is the problem of resolving references to earlier or later items in the text. These items are usually noun phrases representing objects called referents but can also be verb phrases, whole sentences or paragraphs. Anaphora resolution is the standard problem in NLP.

Biotope texts we work with contain several types of anaphora:

- hyperonym definite expressions ("bacterium" - "organism", "cell" - "bacterium"),
- higher level taxa often preceded by a demonstrative determinant ("this bacteria", "this organism"),
- sortal anaphors ("genus", "species", "strain").

For anaphora detection and resolution we use a pattern-based approach. It is based on fixing anaphora items in texts and establishing relations between these items and the objects in conceptual models we use. These objects are bacteria names for "Entity" context, mentions of water, soil and other environment parameters for "Areal" context and names and characteristics of diseases for "Pathogenicity" context.

Corpus tagging is also used for anaphora detection. In particular encoding bacteria (for instance bacterium *Burkholderia phytofirmans* is encoded as PsJN) is found from tagging and further used as its name in text processing.

Noise is constituted by the text elements that contain no facts or cannot be interpreted as facts. Also noise consider the data that are deliberately excluded from

consideration, for example, information about when and by whom a bacterium was first identified.

4.2 **Data Processing**

We have selected 130 mostly known bacteria and processed corresponding corpus texts about them. Three formal contexts of "Entity", "Areal" and "Pathogenicity" had built on the texts. They have the names of bacteria as objects and corresponding concepts from conceptual graphs as attributes.
Table 1 shows numerical characteristics of created contexts.

Context name	Number of objects	Number of attributes	Number of formal concepts
Entity	130	26	426
Areal	130	18	127
Pathogenicity	130	28	692

Table 1. Numerical characteristics of created contexts

As it is followed from the table there is relatively small number of formal concepts in the contexts. This is due to the sparse form of all contexts generated by conceptual graphs and noise reduction.

4.3 **Fact extraction**

Extracting facts from concept lattices is realized by forming special views constructed on the lattice and corresponded to certain property (intent in the lattice) or entity (extent in the lattice) on the set of bacteria. Every view is a sub lattice. It shows the links between concrete bacterium and its properties.

An example of such view as the fragment of lattice is shown on Fig. 3. The lattice on the Fig. 3 contains formal concepts related to the following bacteria: *Borrelia turicatae, Frankia, Legionella, Clamydophila, Thermoanaerobacter tengcongensis, Xanthomonas oryzae*. Highlighted view on the figure corresponds to gram-negative property of bacteria. Such bacteria are resistant to conventional antibiotics.

Using this view, some facts about bacteria can be extracted:

- only three bacteria from the set, *Thermoanaerobacter tengcongensis, Clamydophila* and *Xanthomonas oryzae*, are gram-negative;
- two gram-negative bacteria, *Thermoanaerobacter tengcongensis* and *Xanthomonas oryzae*, have the shape as rod;
- one of gram-negative bacteria, *Clamydophila*, is obligately pathogenic.

Note that attribute *obligately pathogenic* was formed directly from the same two words in the text according to the rule of marking words denoting extreme situation.

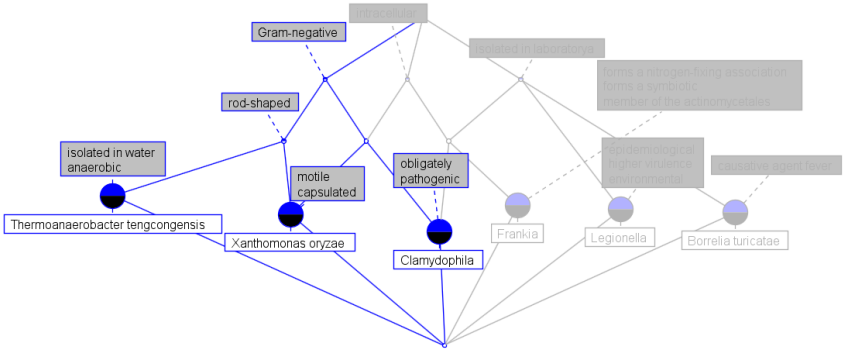


Fig. 3. Example of view concerned gram-negative property of bacteria.

We can compare our results with the known ones, most completely presented in the work [20]. Although we use the same corpus and some resembling methods (they use pattern-based approach and domain lexical resources) our results are different in fact. Our main result is not certain words extracted from texts as solution of NER and RE tasks but the whole information resource of concept lattice which is similar to ontology. So we resume that CGs – FCA modeling provides solving wider set of tasks than Named Entity Recognition and Relations Extraction, the set which corresponds to fact extraction problem.

5 Conclusion and Future Work

This paper describes the first but very important stage of creating environment for performing experiments of CGs – FCA modeling in the project of creating fact extraction technology on natural language texts. Some parts of this project are under construction but current results demonstrate effectiveness of CGs – FCA modeling.

Conceptual graphs were recognized as valid low level conceptual model for creating high level such model of concept lattice. Using conceptual graphs, it is possible to control semantic depth of representing sentences in formal concepts by selecting certain levels (sub graphs) of graph structure.

Among the topics of our future work there are the following.

Now the verb-centric approach which we use in acquiring conceptual graphs is not fully applied for creating formal contexts. When conceptual graph has the pattern <concept> - (agent) – <verb> – (patient) - <concept> the verb serves as condition which links two concepts. In other patterns with other conceptual relations including attribute verbs play the same role. This opens the need to construct tricontexts on conceptual graphs. We plan to construct multidimensional data model on our database under SAP PowerDesigner™ CASE technology and apply OLAP for modeling tricontexts and triclusters.

We also plan to use SAP HANA Environment [21] for work with big textual data.

Acknowledgments. The paper concerns the work which is partially supported by Russian Foundation of Basic Research, grant № 15-07-05507.

References

1. BioNLP 2014. Workshop on Biomedical Natural Language Processing. Proceedings of the Workshop. The Association for Computational Linguistics. Baltimore, 2014. 155 p.
2. Birkhoff, G.: Lattice Theory. Providence, RI: Amer. Math. Soc. (1967)
3. Bogatyrev, M. Y., Vakurin V. S. Conceptual Modeling in Biomedical Data Research. Mathematical Biology and Bioinformatics. 2013. Vol. 8. № 1, pp. 340–349. (in Russian).
4. Bossy R, Jourde J, Manine A-P, Veber P, Alphonse E, Van De Guchte M, Bessières P, Nédellec C: BioNLP 2011 Shared Task - The Bacteria Track. BMC Bioinformatics. 2012, 13: S8, pp. 1-15.
5. Boytcheva, S. Dobrev, P. Angelova, G.: CGExtract: Towards Extraction of Conceptual Graphs from Controlled English. Lecture Notes in Computer Science № 2120, Springer Verlag (2001)
6. Carpineto, C., Romano, G. Using Concept Lattices for Text Retrieval and Mining. In B. Ganter, G. Stumme, and R. Wille (Eds.), Formal Concept Analysis: Foundations and Applications. Lecture Notes in Computer Science 3626, pp. 161-179. Springer-Verlag, Berlin, 2005.
7. Cimiano, P. Hotho, A. Staab, S. Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis. Journal of Artificial Intelligence Research, Volume 24, pp. 305-339. 2005.
8. ConExp-NG. https://github.com/fcatools/conexp-ng
9. Galitsky, B., Dobrocsi, G., de la Rosa, J.L. Kuznetsov, S.O. From Generalization of Syntactic Parse Trees to Conceptual Graphs. In: M. Croitoru, S. Ferre, D. Lukose, Eds., Conceptual Structures: From Information to Intelligence, Proc. 18th International Conference on Conceptual Structures (ICCS 2010), Lecture Notes in Artificial Intelligence (Springer), vol. 6208, pp. 185-190, 2010.
10. Ganter, B., Stumme, G., Wille, R., eds., Formal Concept Analysis: Foundations and Applications, Lecture Notes in Artificial Intelligence, No. 3626, Springer-Verlag. 2005
11. Gildea D., Jurafsky D.: Automatic labeling of semantic roles. Computational Linguistics, 2002, v. 28, 245-288. (2002)
12. Meštrović, A. Semantic Matching Using Concept Lattice. Concept Discovery in Unstructured Data, CDUD 2012, pp. 49-58.
13. Michael Bogatyrev and Alexey Kolosoff. Using Conceptual Graphs for Text Mining in Technical Support Services. Pattern Recognition and Machine Intelligence. - Lecture Notes in Computer Science, 2011, Volume 6744/2011, p.p. 466-471. Springer-Verlag, Heidelberg, 2011.
14. Michael Bogatyrev, Vadim Nuriahmetov. Application of Conceptual Structures in Requirements Modeling. – Proc. of the International Workshop on Concept Discovery in Unstructured Data (CDUD 2011) at the Thirteenth International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing - RSFDGrC 2011. Moscow, Russia, 2011, pp. 11-19.
15. Montes-y-Gomez, M., Gelbukh, A., Lopez-Lopez, M.: Text Mining at Detail Level Using Conceptual Graphs. Lecture Notes In Computer Science; vol. 2393, pp. 122 – 136 (2002)

16. Otero P. G., Lopes G. P., Agustini, A., Automatic Acquisition of Formal Concepts from Text. Journal for Language Technology and Computational Linguistics. Vol. 23(1), pp. 59-74. 2008.
17. Poelmans J., Kuznetsov S. O., Ignatov D. I., Dedene G. Formal Concept Analysis in knowledge processing: A survey on models and techniques // Expert Systems with Applications. 2013. Vol. 40. No. 16. P. 6601-6623.
18. PowerDesigner 16.2. Sybase documentation, DC 00121-01-1520-01. February 2015.
19. Priss, U., Linguistic Applications of Formal Concept Analysis. In: Ganter; Stumme; Wille (eds.), Formal Concept Analysis, Foundations and Applications. Springer Verlag. LNAI 3626, p. 149-160. 2005.
20. Ratkovic, Z., Golik, W., Warnier, P. Event extraction of bacteria biotopes: a knowledge-intensive NLP-based approach. - BMC Bioinformatics 2012, 13, (Suppl 11): S8, pp. 1-11.
21. SAP HANA Environment. https://hana.sap.com/abouthana.html
22. Sowa, J.F., Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks Cole Publishing Co., Pacific Grove, CA, (2000).
23. Wille, R.: Conceptual Graphs and Formal Concept Analysis. Proceedings of the Fifth International Conference on Conceptual Structures: Fulfilling Peirce's Dream. 290 - 303. Springer-Verlag, London. (1997)

Annotated Suffix Trees for Text Clustering

Ekaterina Chernyak and Dmitry Ilvovsky

National Research University Higher School of Economics
Moscow, Russia
echernyak,dilvovsky@hse.ru

Abstract. In this paper an extension of tf-idf weighting on annotated suffix tree (AST) structure is described. The new weighting scheme can be used for computing similarity between texts, which can further serve as in input to clustering algorithm. We present preliminary tests of using AST for computing similarity of Russian texts and show slight improvement in comparison to the baseline cosine similarity after applying spectral clustering algorithm.
Keywords: annotated suffix tree, clustering, similarity measure

1 Introduction

The text clustering applications exploit two major different clustering approaches: either a text is represented as a vector of features, and distance-based algorithms (such as k-Means) are used, or the similarity between texts are computed and the similarity-based algorithms (such k-Medoids or Normalized cuts) are used. While the former requires to extract features from texts, the latter requires the definition of similarity measure. The other algorithms, such as Suffix Tree Clustering [8] explore internal features for the text collection and find clusters straightforward. We will concentrate of the preliminary step of applying distance-based algorithms, i.e. computing similarity between texts. According to [2], there are several approaches to computing text similarity: there are string-based (which include character-based and term-based), corpus-based and knowledge-based approaches. This project belongs to the characters-based approach, which means, we do not take corpora data or semantics into account. We describe a new similarity measure, which is based on the notion of an annotated suffix tree and present an example of using this measure.

2 Annotated suffix tree

The suffix tree is a data structure used for storing of and searching for strings of characters and their fragments [3]. An annotated suffix tree (AST) is a suffix tree whose nodes are annotated by the frequencies of the strings fragments. An algorithm for the construction and the usage of AST for spam-filtering is described in [6]. Some other applications are described in [4, 5].

2.1 Definition of AST

According to the annotated suffix tree model [4–6], a text document is a set of words or word n-grams, which we will address as strings. An annotated suffix tree is a data structure used for computing and storing all fragments of the strings and their frequencies (see Fig. 1 for an example of the AST for the string "mining"). It is a rooted tree in which:

- Every node corresponds to one character
- Every node is labeled by the frequency of the text fragment encoded by the path from the root to the node.

.

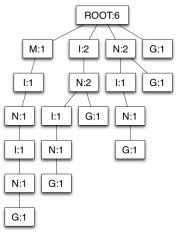


Fig. 1. AST for string "mining"

The AST has two important proprieties: the frequency of a parent node is equal to:

1. the sum of the frequencies of children nodes;
2. the sum of the frequencies of underlying leaves.

According to these properties we can calculate the frequency of the root: it is equal to the sum of the frequencies on the first level of the AST. For example, the frequency of the root of the AST in Fig. 1 is equal to 2+2+1+1 = 6.

2.2 Similarity measure

To estimate the similarity between two texts we find the common subtree of the corresponding ASTs. We do the depth-first search for the common chains of nodes that start from the root of the both ASTs. After the common subtree is constructed we need to annotate and score it.

We annotate the common subtree in the following way. A new node of a common subtree is annotated by two numbers:the minimum and the maximum frequencies of the corresponding nodes of original ASTs.

Let us provide an example of common subtree construction and annotation. Given two ASTs:

– for two strings “mining” and “dining” in Fig. 2
– for one string “dinner” in Fig. 3

we construct the common subtree for them. There are three common chains, which start from the roots: “D I N”, “I N”, “N”. All the nodes of the chain “D I N” have frequencies equal to unity in both ASTs, so in the common subtree the minimum and the maximum frequencies of all three nodes coincide and are equal to unity. The chain “I N” occurs once in the AST in Fig. 3 and four times in the AST in Fig. 2, hence the minimum frequencies are equal to unity and the maximum frequencies are equal to four. The node “N” is annotated by four in the AST in Fig. 2 and by two in the AST in Fig. 3. So its minimum and maximum frequencies are equal to two and four, respectively.

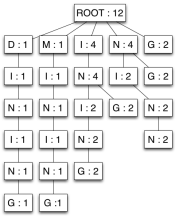


Fig. 2. AST for strings “mining” and “dining”

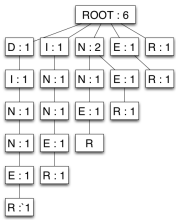


Fig. 3. AST for string “dinner”

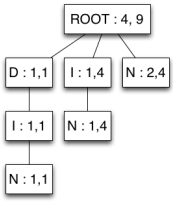


Fig. 4. Common subtree of ASTs in Fig. 2 and Fig. 3

Following the general principle of the AST-based string to text scoring we suggest to score the common subtree in several steps:

1. weighting each node by computing the mean between two frequencies. At this step we can use different type of means, such as geometric mean or harmonic mean. For the sake of simplicity we use further the arithmetic mean;
2. scoring every chain of the common subtree;
3. summing up all chain scores and standardizing them by dividing by the number of chains;

To score the chain we compute the sum of the node frequencies divided by their parents frequencies, which is again divided by the length of the chain:

$$\texttt{score}(chain) = \frac{\sum_{node \in chain} \frac{f^{node}}{f^{parent}}}{|chain|} = \frac{\sum_{node \in chain} \frac{(f^{node}_{min} + f^{node}_{max})/2}{(f^{parent}_{min} + f^{parent}_{max})/2}}{|chain|}$$

For example, the scoring of the common subtree in Fig. 4 is computed as:

$$\frac{\texttt{score}(\text{``D I N''}) + \texttt{score}(\text{``I N''}) + \texttt{score}(\text{``N''})}{3},$$

where

$$\texttt{score}(\text{``D I N''}) = \frac{\frac{(1+1)/2}{(4+9)/2} + \frac{(1+1)/2}{(1+1)/2} + \frac{(1+1)/2}{(1+1)/2}}{3} = \frac{\frac{2}{13} + 1 + 1}{3} = 0.718$$

$$\texttt{score}(\text{``I N''}) = \frac{\frac{(1+4)/2}{(4+9)/2} + \frac{(1+4)/2}{(1+4)/2}}{2} = \frac{\frac{4}{13} + 1}{2} = 0.6538$$

$$\texttt{score}(\text{``N''}) = \frac{\frac{(2+4)/2}{(4+9)/2}}{1} = \frac{6}{13} = 0.4615$$

and the final scoring is:

$$\frac{0.718 + 0.6538 + 0.4615}{3} = 0.6111$$

At this point the scoring of the common subtree is based on using only the frequencies of the strings and their fragments. To make the scoring analogous to computing *tf-idf* we can introduce the *idf*-like component to the scoring.

Let us think about a collection of texts. As a source for *idf* values we can construct a global AST from the whole text collection. To construct the global AST we split every text in strings and exclude from further computations repeating strings and construct the AST then from these unique strings. This way we calculate not the frequencies of the strings, but the number of texts where every string and their fragments occur, that is exactly the *df*'s of all the possible fragments of the texts.

To combine common subtrees and the global tree, we update the chain scoring step:

$$\texttt{score}(chain) = \frac{\sum_{node \in chain} \frac{f^{node}}{f^{parent}} \times \frac{df^{node}}{df^{parent}}}{|chain|},$$

where *df* are extracted from the global tree.

2.3 Construction of AST

It is possible to construct an AST straightforward form a set of tokens using quite an obvious iterative algorithm, which requires splitting each token in suffixes and adding them consecutively to the AST [5]. However, is is shown in [1], that it is more time- and space-efficient to construct a suffix tree, using one of the well-known algorithms and than annotate the tree with the frequencies.

3 Evaluation

We manually created the text collection for further testing of the proposed algorithm. The collection consists of 50 documents in Russian, every 10 text devoted to different definitions of the word "jaguar": an animal, a car, a beverage, a film or a sewing machine. We supposed, that the clusters we would achieve should coincide with the predefined text classes, i.e. we should get five clusters, every cluster corresponding to the initial class. We used the Shi-Malik algorithm [7] with the default parameters to cluster the similarity matrices. Two approaches to the similarity matrix construction:

1. the tf-idf transformation and the cosine similarity
2. the AST technique, presented above.

To compare these approaches we computed the number of errors in the achieved clusters. Given a cluster we found the mode value of the class and calculated how many documents do not belong to this class. The higher this number is, the worse is the result of clustering. Using the cosine similarity and Shi-Malik algorithm for finding five clusters, we achieved four perfect cluster and one cluster, that contained six errors. Hence 44 texts were clustered correctly and 6 were not. Using the AST technique, we got only 2 errors, which means that 48 texts were clustered correctly. The results of clustering are presented in Table 1.

Fig. 5 and Fig. 6 present the heat maps of both similarity measures and reveal some issues of the suggested AST technique. First of all, when the AST technique is applied to compute the similarity of a text to itself, the result is not equal to unity. What is more, for different texts it results in different values. Second, all the similarity values are more or less the same, there is no drastic difference at all between the inside or outside classes values. These are the issues to be solved in the future.

Table 1. Clustering quality

	Accuracy	# of errors
cosine similarity	0.88	6
AST similarity	0.96	2

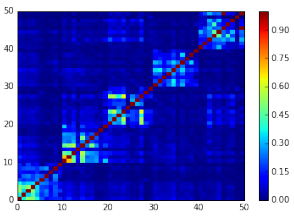


Fig. 5. Heat map of the cosine similarity matrix

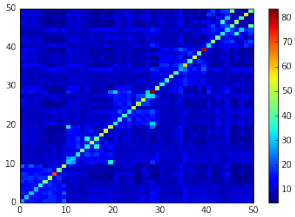


Fig. 6. Heat map of the AST similarity matrix

4 Conclusion

We suggest a new text similarity measure, which is based on annotated suffix trees. To estimate the similarity between text A and text B, one should construct two annotated suffix trees, find the common subtree and score it. The scoring can be extended by document frequencies, if a text collection is given. The preliminary experiments show, that although the proposed similarity measure has some clear advantages in comparison to the baseline cosine similarity, because of being more robust, some formal aspects should be improved. For example, currently the similarity of a text to itself is not equal to unity, which affects the visualisation. Obviously, more experiments should be conducted to find other limitations and possible improvements.

Acknowledgments The paper was prepared within the framework of the Basic Research Program at the National Research University Higher School of Economics (HSE) and supported within the framework of a subsidy by the Russian Academic Excellence Project "5-100". The authors was partially supported by Russian Foundation for Basic Research, grants no. 16-29-12982 and 16-01-00583.

References

1. Dubov, M.: Text Analysis with Enhanced Annotated Suffix Trees: Algorithms and Implementation. In Analysis of Images, Social Networks and Texts, 2015, pp. 308-319. Springer International Publishing.
2. Gomaa, W. H., Fahmy, A. A.: Survey of text similarity approaches. International Journal of Computer Applications, 2013, 68(13).
3. Gusfield D.: Algorithms on Strings, Trees, and Sequences, Cambridge University Press, 1997.
4. Chernyak E.L., Chugunova O.N., Askarova J.A., Nascimento S., Mirkin B.G., Abstracting concepts from text documents by using an ontology, in Proceedings of the 1st International Workshop on Concept Discovery in Unstructured Data. 2011, pp. 21-31.
5. Chernyak E.L., Chugunova O.N., Mirkin B.G.: Annotated suffix tree method for measuring degree of string to text belongingness, Business Informatics, 2012. Vol. 21, no.3, pp. 31-41 (in Russian).

6. Pampapathi R., Mirkin B., Levene M.: A suffix tree approach to anti-spam email filtering, Machine Learning, 2006, Vol. 65, no.1, pp. 309-338.
7. Shi, J., Malik, J.: Normalized cuts and image segmentation. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2000, 22(8), 888-905.
8. Zamir, O., Etzioni, O.: Web document clustering: A feasibility demonstration. In Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval (pp. 46-54). 1998, ACM.

Single-Focus Broadening Navigation in Concept Lattices

Gillian J. Greene and Bernd Fischer

CAIR, CSIR Meraka
Computer Science Division
Stellenbosch University, South Africa
{ggreene, bfischer}@cs.sun.ac.za

Abstract. Formal concept analysis has been used to support information retrieval tasks in many domains, in particular the traditional "by keyword" document search with a conjunctive query interpretation. However, support for exploratory search or *browsing* needs new navigation algorithms that allow users (*i*) to continuously update the current query and (*ii*) to broaden as well as refine the result set. In this paper we investigate a step-wise navigation algorithm that supports both broadening and refinement operations. Our navigation operations maintain some useful algebraic properties. We motivate our approach on a dataset of wine reviews, which contains different facets of information.

Keywords: Information Retrieval, exploratory search, step-wise navigation, broadening navigation, Formal Concept Analysis

1 Introduction

Formal concept analysis has been used to support information retrieval (IR) [1, 2] tasks in many domains [3, 4] and to implement different IR algorithms. The traditional approach [5] to IR using formal concept analysis views the documents as objects and their associated meta-data and extracted terms as attributes. The concept lattice is then computed from these document contexts. Each concept's intent represents a possible query (interpreted as the conjunction of all corresponding terms), with the extent forming the set of retrieved documents [5].

One particular IR task, and the one that we are interested in, is exploratory search [6] or *browsing* [7, 8]. It is aimed at familiarizing the user with the underlying data through serendipitous navigation, and so complements the traditional, direct keyword lookup-based document retrieval. Browsing is supported in graph structures by moving from vertex to vertex where each vertex represents the current query [7]. Therefore, in order to implement browsing with concept lattices, we need a step-wise navigation algorithm that allows users (*i*) to incrementally update the current query and (*ii*) to restrict (i.e., move down in the lattice) as well as broaden (i.e., move up in the lattice) the result set. In this paper we focus on such step-wise navigation algorithms and in particular a broadening navigation approach.

Concept lattices can in principle be navigated directly, by following the subconcept relation to move from one concept to another concept in its direct neighborhood [7]. However, this only allows for small navigation steps and thus restricts the serendipitous nature of the browsing operation and becomes impractical for large lattices. Instead, we aim at large step navigation algorithms that allow users to select and deselect arbitrary attributes and rely on the meet and join operations to move between concepts [9].

Large-step navigation algorithms should ideally satisfy a number of properties that ensure that their behavior is transparent to users. First, they should be *Markovian*, i.e., rely only on the current query concept and the new selection (or de-selection) to determine the next concept as result of the navigation step. This means that users do not need to remember the navigation history in order to understand the results. Second, they should be *Abelian*, i.e., the order of the navigation steps should have no effect on the next navigation result. This allows users a certain degree of freedom in how they navigate through the underlying document collection. Finally, they should have the *single focus* property, i.e., each query result can be represented by a single concept in the lattice. If the concept lattices constructed from the document contexts were Boolean lattices then these properties would follow automatically; however, this is not the case for most document collections.

If we follow a purely conjunctive query interpretation (i.e., consider all query terms to be connected by the AND operator), we can use the lattice's meet operation as implementation of the AND operator [10] in a document-term concept lattice. Moreover, the navigation algorithm is then by construction Markovian and Abelian, and has the single focus property. However, this does not provide us with disjunctive queries, or, with any broadening navigation operations. We therefore investigate broadening navigation approaches that maintain only a single focus concept.

It remains unclear what exactly constitutes broadening navigation, and there are several different operations that extend the query result and can be considered as "broadening".

- We can de-select a previously selected term; under a purely conjunctive query interpretation the new focus is then computed as the meet of the introducing concepts of the remaining terms (although Lindig [11] has described an optimized implementation). Note that the new focus has not necessarily been visited during the previous navigation steps (so de-selection is not always an undo operation), but it is a super-concept of the old focus, and conjunctive navigation with selection and de-selection is still Markovian and Abelian.
- We can use a separate concept to represent each argument of an OR operator; the result of such a disjunctive query is then the union of all corresponding extents [12]. However, this disjunctive navigation gives up the single focus property and is no longer Abelian, since the order of AND and OR operators matters.
- We can also retrieve or insert a *query concept* [13] into the lattice, where the *query concept's* intent contains the current search terms and retrieve the

	{country}		{review text}		{varietal}		
	USA	South Africa	Fruity	Berry	Cabernet Sauvignon	Merlot	Pinotage
wine-bottle1	X		X		X		
wine-bottle2		X	X	X	X		
wine-bottle3	X			X		X	
wine-bottle4		X		X			X

Fig. 1: Example context derived from wine review data, wine bottles are objects with review text, review year, vintage, location and winery as attributes. Attribute facets are indicated by the color of the column.

parents of the query concept (called the *query generator*), as a generalization [14]. Additionally, more children of the query generator can be included to broaden the results further. These are referred to as *cousin concepts* [14].

– We can use the lattice's join operation as a generalization operation; if the generalized concepts are determined by objects (rather than attributes) this is also known as *object-based navigation* [15]. This navigation has the single focus property by construction, and is still Markovian and Abelian (when it is not mixed with refinement operations, otherwise lattice distributivity is also required to ensure the Abelian property), but does not implement the Boolean OR operation: due to the closure operations in the lattice construction, the extent of the new focus typically contains additional objects. This could be seen as a feature [15] but in contexts where the attributes represent different categories or *facets* [16] this is prone to *overgeneralization*. Overgeneralization refers to the focus moving too high in the lattice (possibly to top) which would result in a decrease of precision for the query's results. In particular, if we have functional facets (where each object can have only a single attribute for a given category, such as year of birth), the join will effectively cancel the selected attributes from this category. The join operation is thus unsuited as an intuitive generalization operation. We therefore investigate an alternative generalization operation that makes use of only subsets of the extents of the attribute concepts of the selected items, in order to provide a more intuitive broadening navigation.

Our approach is motivated from navigation in a dataset of wine reviews extracted from [17]. The full dataset contains over 16000 objects. However, we use a small example of the dataset in order to make the drawing of the concept lattices feasible. For each wine bottle we have as attributes, the winery, the vintage, the reviewer, the review year as well as the location and keywords extracted from the reviews. We use individual wine bottles as objects in the context and assign all other fields as the attributes. Figure 1 provides an example of the constructed context for this dataset. This dataset contains functional facets, such as the country, where each wine bottle can originate from only one country.

In this paper we provide a brief overview of information retrieval tasks and navigation in concept lattices (Section 2). We then illustrate our refinement selection and de-selection approach (Section 3) where the de-selection operation reverses a refinement selection. We define a single-focus boolean OR navigation operator, followed in Section 5 by an intuitive generalization operation which prevents the full object set in the lattice from being returned. Additionally in Section 5 we discuss an approach for finding similar objects within the concept lattice.

2 Information Retrieval and Navigation using Concept Lattices

There have been many approaches to supporting information retrieval tasks using concept lattices, some of which extend to disjunctive queries and broadening approaches.

Codocedo et al. [14] propose an information retrieval approach using concept lattices where queries are answered using the *cousin concepts* of the *query concept*. The *query concept* is inserted into (or identified in) the concept lattice with a placeholder object and all the attributes that form a part of the current query [13]. The superconcept of the query concept is then referred to as the *query generator*. The *cousin concepts* of the query concept refer to the subconcepts of the *query generator*. The cousin concepts and the query generator are used to implement a broadening approach in the concept lattice [14]. The query's result is then returned as the union of the cousin concepts' extents.

Ferré [18] uses a navigation technique where a generalization is similar to our de-selection (it does not need to take place in any particular order) and de-selection refers only to removing the last selected item (e.g., an undo operation).

Godin et al. [8] described an iterative retrieval algorithm which maintains a *focus* concept whose extent is the retrieval result. Initially, the focus is the lattice's top element; in each iteration the user moves it to an adjacent concept, by adding (removing) an attribute (not) in the intent of a concept directly above (below) the current focus. However, this navigation style is too incremental, because the focus can move only one level at a time, and too constrained, because the user can only choose attributes from the intents of the directly adjacent concepts, and has no indication which choices are hidden behind paths not taken.

Lindig [9] introduced a semi-constrained navigation algorithm where, the focus can be refined by selecting *any* attribute from *any* concept (except $\perp$) below the focus, provided the attribute is not already in the focus' intent. The focus is then updated by computing its meet with the attribute concept. A restriction on selectable attributes prevents navigation into dead ends, and ensures that each query refinement also refines the query results.

Fischer [15] exploited the duality of concept lattices and introduced object-based navigation; here, selection of an object not in the focus' extent is a widening step that is implemented via the join.

Lindig [11] uses object-based navigation to implement *relevance feedback*; selecting an object in the focus' extent selects all attributes in the intent of its object concept.

3 Refinement Selection and Deselection

Refinement operations in the lattice can be computed using the meet operation. For step-wise navigation, we maintain a current focus concept at each navigation step. The focus can be refined with a new selection by calculating the meet of the current focus and the attribute concept of the new selection.

Additionally, items available for selection can be restricted to those that have a non-bottom meet with the focus, ensuring that a selection never returns an empty extent.

Because of the duality in the lattice, we might expect that the de-selection (removing a previously selected item) can be implemented by the join (least upper bound) operation, however this is not the case. Intuitively the de-selection of the most recently selected item should return the focus concept to its previous position, undoing the selection. However, computing the join of the focus with the attribute concept of the new de-selection will cause all previous selections to be removed, except the attribute we are de-selecting, which is counterintuitive. Therefore, in order to reverse a single selection operation we need to recompute the focus as the meet in the lattice from all *still-selected* items.

Our de-selection performs essentially the same operation as illustrated by Lindig [11], although Lindig optimizes this operation by making use of the search path in order to compute the new focus concept. Note that de-selections do not always need to take place in the same order as the initial selections. The de-selection operation can return a focus which has not been visited during the previous navigation steps. De-selection is therefore not only an undo operation.

4 Boolean Disjunctive Selection

The meet operation in the lattice satisfies conjunction between selected items. The meet of the attribute concept of item a ($\mu(a)$) AND the attribute concept of item b ($\mu(b)$), results in a concept whose intent contains both items a AND b. However, boolean OR navigation, where an attribute must only apply to *at least one* object is not supported by either the meet or join operations.

Priss [12] makes use of a boolean disjunctive query operation which returns the union of the extents of the concepts that are retrieved for each of the items in the query when selected individually. This approach requires more than one focus to generate the query's result.

Codocedo et al.'s approach [14] (illustrated in Figure 4) does not implement a purely disjunctive query operation as the query generators are not necessarily the attribute concepts of the items selected for the disjunctive query.

In our approach (Figure 3) we alter the underlying context table in order to support the disjunctive navigation and maintain the single-focus property. By

updating the underlying context we are able to support further navigation steps (refinements or generalizations) and maintain the disjunctive queries using only a single focus concept.

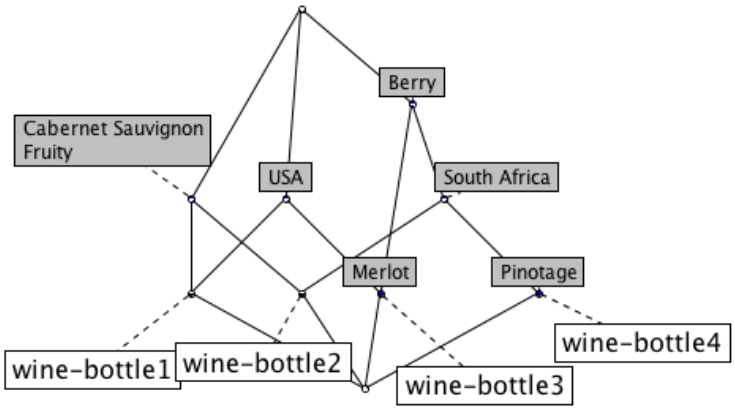


Fig. 2: Full concept lattice generated from the wine data context in Figure 1. The attributes are generated from different facets such as wine varietals and the objects are the individual wine bottles. Lattice generated with Concept Explorer [19]

In order to compute the boolean OR of two items, a and b, we merge the items in the underlying context table into a new attribute $a\ OR\ b$. We compute the introducing concept of the newly created merged item ($\mu(a\,or\,b)$) as the new focus. Our approach therefore returns the same query results as those that would be obtained in [12] for a *single* disjunctive query with no consequent navigation steps.

Figures 2 and 3 illustrate our approach. Figure 2 shows the initial concept lattice generated from the unaltered context. However, if we select item "Cabernet Sauvignon", the focus will be the meet of $\top$ and the attribute concept of "Cabernet Sauvignon" (resulting in the attribute concept of "Cabernet Sauvignon"). Selecting "Merlot" for a boolean OR operation, the context will be updated to add the combination of the two attributes to the context and the updated lattice will appear as in Figure 3.

The join of the focus and the attribute concept of "Merlot" would return the top concept in the lattice and our result-set would contain wines of other varietals (such as "Pinotage") which is undesirable. By using the boolean OR operation we are able to retrieve all wines that are only of the "Merlot" OR "Cabernet Sauvignon" varietals and the attributes which these wines possess.

Note that this approach is similar to the use of conceptual scales in the concept lattice [20] for multi-valued attributes (such as prices). However, instead of using pre-defined scales, our scale is generated automatically when the user

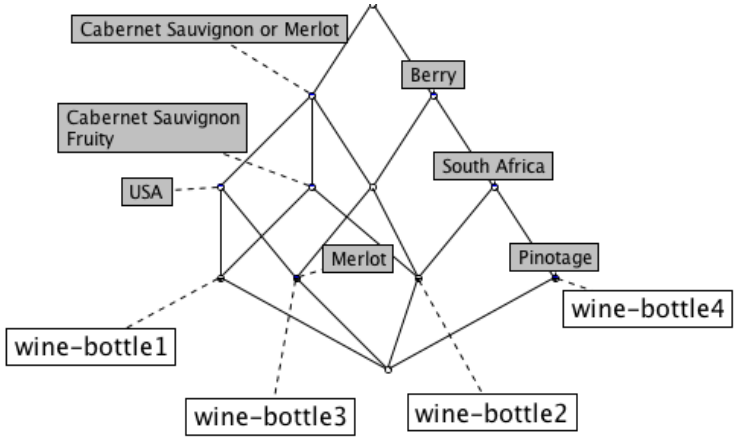


Fig. 3: Full concept lattice generated from the wine data context in Figure 1; where attributes "Cabernet Sauvignon" and "Merlot" have been selected for boolean OR navigation. A new concept with attribute "Cabernet Sauvignon or Merlot" has been inserted into the lattice. Lattice generated with Concept Explorer [19]

makes a boolean OR selection of an item in the dataset. The scale is therefore interactively created and we update the context on-the-fly.

5 Broadening Navigation Approach

The join operation supports broadening navigation, however, if the extents of both concepts are large then the join is likely to overgeneralize and can result in the top concept ($\top$) thereby losing all previous navigation steps and resulting in a low precision for the constructed query.

In order to support a broadening selection, that does not overgeneralize and return a concept with a large extent (and little or no common attributes) resulting in a low precision, we compute the join from only a *subset* of the objects in the full extents of the two concept selections. If the join of the two concepts is not $\top$ then we return their join, otherwise we recompute the join after removing one or more objects, until the join does not result in the top concept.

5.1 Generating Candidate Focus Concepts

Our broadening approach results in an updated focus that shows attributes which are common to *some* of the objects in the current focus *and some* of the objects in the attribute concept of the new item ($\mu(b)$) selected for broadening.

For example, in our wine review dataset if we select winery a for refinement and then broaden on winery b, our updated focus will show which wine characteristics (text from reviews etc.) are common to *some* bottles produced at

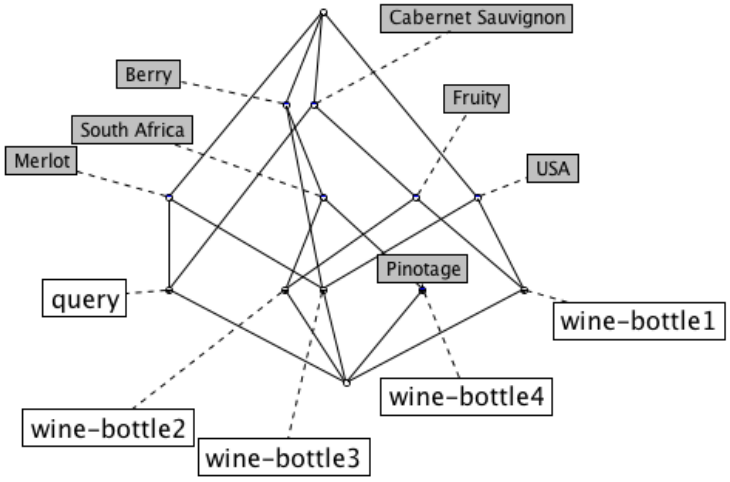


Fig. 4: Query concept with intent "Merlot" and "Cabernet Sauvignon" inserted into the concept lattice. Lattice generated with Concept Explorer [19]

winery b and *some* bottles produced at winery a. The join would reveal only characteristics that are common to *all* wine bottles from winery a and winery b, and since the set of all bottles from winery a and winery b it is likely to be large, the join risks navigating to top ($\top$).

If the join does not return $\top$, we return the join concept as the new focus, otherwise we traverse the lattice with a top-down depth-first approach using the focus as starting point. For each new concept in this traversal we then also perform a top-down depth-first traversal starting at $\mu(b)$. We compute the join of every concept derived from these iterations as candidate focus concepts for the next navigation step as shown in Algorithm 1.

Note that the amount of candidate focus concepts could be large and therefore we need to select a new focus from this pool in order to maintain only a single focus.

5.2 Selecting a new Focus Concept from the Candidate Focus Concepts

We choose a single focus from the generated set of candidate focus concepts. If a join for the previous focus (a = (A, B)) and the introducing concept of the selection (b = (C, D)) that is not the top concept in the lattice exists then we return the join concept, otherwise if the join results in the top concept then we want to return the highest concept (with the largest extent) such that at least one object from concept (a) is present and at least one object from concept (b) is contained in the extent.

```
Data: Current focus concept f, and attribute concept of selected item b, μ(b)
Result: Focus concept candidates for broadening navigation
iterator = top_down_traversal starting at f
while iterator.next is not null do
    f_subset = iterator.next
    inner_iterator = top_down_traversal starting at μ(b)
    while inner_iterator.next is not null do
        μ(b)_subset = iterator.next
        concept = join of μ(b)_subset and f_subset
        add concept to candidate_focus_concepts
    end
end
```

Algorithm 1: Computing candidate focus concepts for our broadening navigation operation. Join operations are computed using only a subset of the objects in the extents of the attribute concepts of the selections.

We therefore return the concept ($c = (E, F)$) which results in the highest score where the score is computed as

$score = |E| - \big||A| - |C|\big|$ where $|A| > 0$ and $|C| > 0$.

Our broadening operation therefore generalizes as much as possible without losing all previous selections and navigation steps and removing all previous navigation steps (navigating to $\top$).

6 Finding Similar Objects

Another method of generalizing from a single object in the dataset is to find a group of related or similar objects. To find objects that are similar to a selected object in the dataset we introduce a *more like this* operation. For example, if we want to find bottles of wine that are similar to a bottle that we have previously tried (i.e. generalize from a single wine bottle), we can apply the *more like this* operation to shift our focus to a concept that contains similar bottles, without the user needing to be aware of any of the attributes of the wine.

All concepts in the lattice in which the object of interest appears in the extent can be considered to present similar objects. However, in multi-faceted data, we are interested in finding a similarity between the objects in comparable facets (e.g, wine bottle 1's origin and wine bottle 2's origin). We also restrict the operation to returning results from only a single concept in the lattice so that all subsequent navigation steps can continue after a *more like this'* generalization operation.

Since not all facets can be used to compare objects, for example being reviewed by the same wine reviewer may not imply that two wine bottles are similar, we use only relevant facets (such as the wine review text and varietal) to compare objects. Various objects in the lattice will be similar across different dimensions. We look at descriptors from relevant facets of the object that are introduced lower in the lattice (are more specific) and include as many of these

as possible in the meet calculation to derive the new focus. However, if specific terms result in the meet returning only the original object of interest then we remove these terms in order to move the focus up and generate a larger extent.

Calculating the size of the extent of the attribute concept can provide a kind of TF/IDF [21] measure for the attribute in the entire corpus of objects. If the extent of an attribute concept is large, then the objects in that extent are unlikely to be very similar, since the attribute can be considered to be less specific as it applies to a large portion of the corpus.

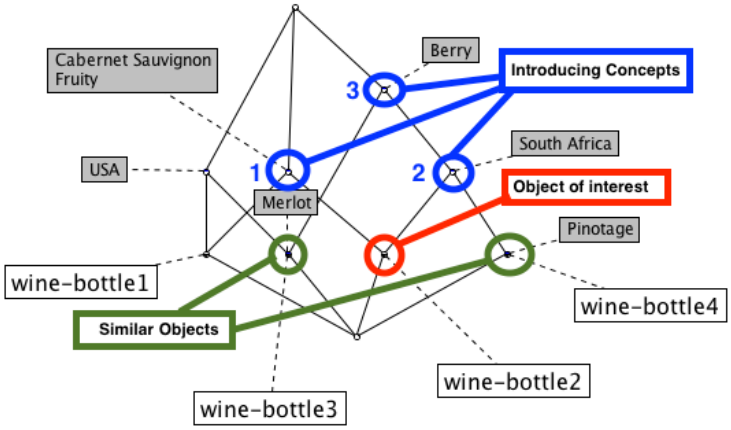


Fig. 5: Finding Similar Objects in a Concept Lattice. The object of interest is indicated in red, its attribute concepts are labeled 1,2 and 3. Similar objects (bottles 3 and 4) are indicated in green.

Figure 5 provides an example of our approach to finding similar objects. Wine bottle 2 (indicated in red) is the object of interest. The introducing concepts of the attributes of wine bottle 2 (berry, south africa, fruity, cabernet sauvignon) are indicated in blue. The meet of all three of these concepts would lead only to wine-bottle 2. Therefore the concept with the smallest extent is removed from the meet set first in order to move the focus up. Since concepts 1 and 2 both have an equal extent size, we use the size of the intent in order to decide which concept to remove from the meet calculation. Concept 1 provides two introducing attributes and so we remove concept 2 from the meet set. The meet of concepts 1 and 3 returns only wine bottle 2 (providing no similar wine bottles) and so concept 1 is subsequently removed from the meet set, leaving only concept 3. Therefore, wine bottles 3 and 4 are considered similar to wine bottle 2 in our approach because according to their reviews they all share flavors of “berry”. Although bottle 1 can also be considered similar to wine bottle 2 as they both share attribute (“Merlot”), our approach favors the more general concept (concept 3) so that the updated focus has a larger extent, including more similar objects.

7 Related Work

There have been many applications of concept lattices in the information retrieval domain [22], for example, the FaIR [12] and Credo systems [5].

The FaIR information retrieval system [12] combines a lattice-based thesaurus approach with boolean queries. The lattice-based thesaurus is used to generate the query language. Terms from each facet are separated into different lattices, unlike in our approach where the term and the facet name are used to represent a term in a single lattice. The thesaurus is used to add synonyms to the lattice so that queries with a wider vocabulary can be handled.

Credo [5] facilitates the exploration of web search results. Index terms from each retrieved search result are extracted from the documents. Credo supports refinement of the search results by selecting additional terms. Initially the presented information is derived from the lattice's top element and possible refinements are presented to the user. These refinement terms can then be selected to display a more specific set of search results and refine the initial query. Credo only includes support for refinement navigation.

8 Conclusions

In this paper we have discussed step-wise refinement and broadening navigation approaches in concept lattices that maintain the single focus property. We have developed a broadening navigation algorithm that makes use of subsets of the extents of two concepts in order to prevent the join from resulting in the top concept in the lattice. We have modified the disjunctive navigation technique to allow only a single focus concept to be stored and used to generate the results of the disjunctive query, allowing consequent broadening and refinement navigation steps to take place. We have discussed refinement navigation in concept lattices and our de-selection operation which is able to reverse refinement selections and does not restrict the order of the de-selection operation. Our navigation approaches can be used to facilitate exploratory search in large concept lattices and allow subsequent refinement, broadening and boolean OR navigation operations to take place.

Acknowledgments

This research is funded in part by a STIAS Doctoral Scholarship, CAIR-SU and NRF Grant 93582.

References

1. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill, Inc., New York, NY, USA (1986)
2. Rijsbergen, C.J.V.: Information Retrieval. 2nd edn. Butterworth-Heinemann, Newton, MA, USA (1979)

3. Poelmans, J., Ignatov, D.I., Viaene, S., Dedene, G., Kuznetsov, S.O., et al.: Text mining scientific papers: A survey on fca-based information retrieval research. In: ICDM. Volume 7377., Springer (2012) 273–287
4. Poelmans, J., Ignatov, D.I., Kuznetsov, S.O., Dedene, G.: Formal concept analysis in knowledge processing: A survey on applications. Expert Syst. Appl. **40**(16) (2013) 6538–6560
5. Carpineto, C., Romano, G., Bordoni, F.U.: Exploiting the potential of concept lattices for information retrieval with credo. J. UCS **10**(8) (2004) 985–1013
6. Marchionini, G.: Exploratory search: from finding to understanding. Communications of the ACM **49**(4) (2006) 41–46
7. Godin, R., Pichet, C., Gecsei, J.: Design of a browsing interface for information retrieval. SIGIR Forum **23**(SI) (May 1989) 32–39
8. Godin, R., Saunders, E., Gecsei, J.: Lattice model of browsable data spaces. Information Sciences **40**(2) (1986) 89–116
9. Lindig, C.: Concept-based component retrieval. In: Working Notes of the IJCAI-95 Workshop: Formal Approaches to the Reuse of Plans, Proofs, and Programs. (1995) 21–25
10. Carpineto, C., Romano, G.: Effective reformulation of boolean queries with concept lattices. In: Flexible Query Answering Systems. Springer (1998) 83–94
11. Lindig, C.: Algorithmen zur Begriffsanalyse und ihre Anwendung bei Softwarebibliotheken. PhD thesis, TU Braunschweig (1999)
12. Priss, U.: Lattice-based information retrieval. Knowledge Organization **27**(3) (2000) 132–142
13. Carpineto, C., Romano, G.: Order-theoretical ranking. Journal of the American Society for Information Science **51**(7) (2000) 587–601
14. Codocedo, V., Lykourentzou, I., Napoli, A.: A semantic approach to concept lattice-based information retrieval. Annals of Mathematics and Artificial Intelligence **72**(1-2) (2014) 169–195
15. Fischer, B.: Specification-based browsing of software component libraries. Autom. Softw. Eng. **7**(2) (2000) 179–200
16. Prieto-Diaz, R.: Implementing faceted classification for software reuse. Communications of the ACM **34**(5) (1991) 88–97
17. : Wine review online. `http://www.winereviewonline.com/`
18. Ferré, S.: Camelis: a logical information system to organise and browse a collection of documents. International Journal of General Systems **38**(4) (2009) 379–403
19. : Concept explorer. `http://conexp.sourceforge.net/`
20. Ganter, B., Wille, R.: Conceptual Scaling. In: Applications of Combinatorics and Graph Theory to the Biological and Social Sciences. Springer US, New York, NY (1989) 139–167
21. Sparck Jones, K.: A statistical interpretation of term specificity and its application in retrieval. Journal of documentation **28**(1) (1972) 11–21
22. Codocedo, V., Napoli, A.: Formal concept analysis and information retrieval–a survey. In: Formal Concept Analysis. Springer (2015) 61–77

Gender Prediction for Authors of Russian Texts Using Regression And Classification Techniques

Tatiana Litvinova[1,3], Pavel Seredin[2,3], Olga Litvinova[1,3], Olga Zagorovskaya[1,3], Alexandr Sboev[3], Dmitry Gudovskih[3], Ivan Moloshnikov[3], Roman Rybka[3]

[1]Voronezh State Pedagogical University, Voronezh, Russia
centr_rus_yaz@mail.ru
[2]Voronezh State University, Voronezh, Russia
paul@phys.vsu.ru
[3]Kurchatov Institute, Moscow, Russia
sag111@mail.ru

Abstract. Automatic extraction of information about authors of texts (gender, age, psychological type, etc.) based on the analysis of linguistic parameters has gained a particular significance as there are more online texts whose authors either avoid providing any personal data or make it intentionally deceptive despite of it being of practical importance in marketing, forensics, sociology. These studies have been performed over the last 10 years and mainly for English. The paper presents the results of the study of a corpus of Russian-language texts *RusPersonality* that addressed automatic identification of the gender of the author of a Russian text using mostly topic-independent text parameters. The identification of the gender of authors of texts was addressed as a classification as well as regression task. For the first time for Russian texts we have obtained the models classifying authors of texts according to their gender with the accuracy identical to the state-of-the-art one.

Keywords: authorship profiling · corpus · stylometry · text classification · regression · gender attribution

1 Introduction

In recent years, exponential increase in textual information has sparked interest in automatically predicting users' personal information (gender, age, personality traits and so on). This field of research is often referred to as authorship profiling. Automatic prediction of such information has various applications in the fields of forensics, business intelligence and security.

The general algorithm for solving this problem is as follows:

1) Collecting a corpus of texts with metadata containing information about their authors;

2) Designing a list of text parameters, linguistic labelling of a corpus and extraction of numerical values of selected text parameters;

3) Designing a mathematical model to detect a certain personality trait based on qualitative values of texts and evaluation of their accuracy.

This area of research has been rapidly developing. There have been contests to find most accurate techniques for categorizing texts according to their authors' personal information [10].

One of the most important characteristics of authors of texts is their gender, i.e. there are a lot of papers on automatic detection of personality traits using texts. Research in identifying author's gender started with extensions of the earlier work on categorization and classification of text [7]. Using the various methods and features, researchers have automated prediction of an author's gender with accuracies ranging from 80% to 90%. [1;2;4;5;12]. For instance, the winners of PAN 2015 obtained models to classify texts according to the gender of their authors with the accuracy as high as 0.97 for Danish and Spanish and 0.86 for English [10].

There are still a lot of issues to be addressed and selecting the parameters to study seems most crucial. Different groups of text parameters were used which can be extracted using NLP tools such as content-based features (bag of words, words n-grams, dictionary words, slang words, ironic words, sentiment words, emotional words) and style-based features (frequency of punctuation marks, capital letters, quotations, together with POS tags) as well as feature selection along with a supervised learning algorithm (see [10] for review). Different research including the one mentioned above have used the parameters of the frequency of words of different topics but it is obvious that the resulting models might not be appropriate to use for corpora of texts of other genres. We are also cautious about the fact that «the reported performance might be overly optimistic due to non-stylistic factors such as topic bias in gender that can make the gender detection task easier» [12, p. 78]. Therefore it is essential that the high-frequency parameters less dependent on a particular topic and genre are used.

Most studies of the classification of texts according to the gender of their authors have been conducted using English texts and there have been only a few studies dealing with other languages (see [3] for details), especially Slavic ones.

The author's gender is known to be explicitly expressed in Russian texts if a verb in a sentence is in the past form and the subject is a singular first-person pronoun "я". Compare: "Прошлой зимой я **ездила** в Альпы" (a female speaker); "Прошлой зимой я **ездил** в Альпы " (a male speaker). If the subject is not the pronoun "я" or if the verb is not in the past form, the gender of the speaker is not explicit. Compare: "Я **поеду** в Альпы" (the gender of the speaker is not explicit). It is worth emphasizing that the existence of grammatical forms which reflect the speaker's gender does not automatically make gender identification in Russian texts a trivial task. In Russian "the gender of the speaker" is explicit in a statistically insignificant number of statements. Any non-first-person narrative does not indicate the gender of its author. Besides, it is easy for the author to imitate the speech of an individual of the other gender using the above forms. Therefore it is only by relying on these parameters that the gender of the author can be identified particularly in a forensic context.

In our lab, we focus on identifying the gender of authors of Russian texts using different methods of data analysis and different sets of text parameters. The basic assumptions of our research rely on issues facing forensic analysis, i.e. we use relatively

short texts (150-300 words) as material for training and testing models and the text parameters were relatively topic-independent and cannot be consciously controlled and imitated and quantifiable and extracted by means of different NLP tools.

2 Design of the study

2.1 Dataset

For this study, we used corpus "RusPersonality" which consists of Russian-language texts of different genres (e.g. description of a picture, essays on different topics, etc.) labelled with information on their authors (gender, age, results of psychological tests, and so on). As of now, the corpus "RusPersonality" contains 1 867 texts by 1 145 respondents (depending on the type of a task, they wrote one or two texts). Overall corpus contains about 300 000 words. The average length of texts was 230 words. Most of the respondents were students of Russian universities. For experimental studies of automatic identification of an author's gender we selected only students' texts so that other factors (age, education level, etc.) do not have any influence on gender and linguistic text parameters. Selections in all of the experiments were balanced by gender. The selections in all the experiments were balanced by gender.

2.2 Feature set

We employed different text parameters that are relatively topic-independent.

Morphological features:

- POS tag features, which mainly represent a particular part of speech for every word in a given text: the number of nouns; the number of numerals; the number of adjectives; the number of prepositions; the number of verbs; the number of pronouns; the number of interjections; the number of adverbs; the number of particles, the number of conjunctions, the number of participles, the number of infinitives, the number of finite verbs (were extracted in different experiments using a morphological parser by XEROX, pymorphy 2 library script, morphosyntactic parser [11]);
- Derivatives of the coefficients which were different relationships of parts of speech: Treiger index, dynamics coefficient, 27 in total [9], [13];
- POS bigrams extracted using a morphological parser by XEROX [8];

1. Syntactical features (60):

- Synto – frequencies of different types of syntactic relationships between heads and dependents. Syntactic structure of sentences was analyzed as a dependency tree and extracted using a morphosyntactic parser [11];
- number of sentences of different types: compound and complex, etc. (extracted manually);

2. *Punctuation features* – the number of commas, exclamatory marks, the number of question marks; the number of dots; the number of emoticons etc. extracted by means of a specially designed script;
3. *Lexical features* – lexical diversity indices extracted using online service istio.com and EmoDicts – frequencies of words denoting different types of emotions (e.g., "Anxiety", "Discontent", the total of 37 categories, see [6] for details).

2.3 Methods

We have addressed automatic detection of an author's gender as a regression and text classification task. Logistic regression was designed using IBM SPSS Statistics software.

Basically, the prediction of gender and age of the author of a text document is made by machine learning algorithms. Independent of the classifier used (see Section IV-D), the input consists of a large list of features with appropriate values and a corresponding classification class. The class is used to train the algorithms if the document is part of the training set, as well as for evaluating if the document is part of the test set. To determine the best working algorithm for this approach, several commonly used methods have been tested, which are well studied and have been used extensively in several text classification tasks. In particular, we used Gradient Boosting Classifier, Adaptive Boosting Classifier (adaBoosting), ExtraTrees, Random Forest, PNN (sigma = 0.1), Support Vector Machine with linear kernel (SVMs), ReLU (1 Hidden Layer with 26 neurons). Python libraries were used for learning the classification models: scikit-learn fitted with machine learning methods and keras for designing neural network models (http://scikit-learn.org/, https://pypi.python.org/pypi/Keras).

3 Results

3.1 Regression models

1. First experiment. For a pilot study, 150 texts from 75 participants (26 males, 49 females) were selected with the average number of words being 166. There was a total of 75 text parameters all of which are relatives values that is correlations of numerical values of different text parameters (part-of-speech correlations, e.g. (vfin+vinf)/noun, adj/(adv+pronadv), correlations of the number of types of various syntactic structures and so on) [9]. Ratios, i.e. relative frequencies, were used as the parameters in order to refrain from the dependence on the length of the text.

In order to estimate the closeness and direction of the linkage between the parameters of the text and personality and to establish the analytical expression (form), correlation and regression analysis was used based on modern statistical data visualization software. The main aim of the study was to establish a function dependence of a conditional mean of the result property (Y) (gender) on the factor properties (x_1, x_2, …, x_k), which are the parameters of the text. Therefore the initial regression equation, or a statistical model of the relationship between the author's gender and quantitative parameters of the text is given by the function

$$Y(x) = f(x_1, x_2, \ldots, x_n), \quad (1)$$

where n is a number of factors included in the model; x_i are the factors that influence the result Y.

In order to determine the characteristics of and type of connection between the text parameters and individual characteristics of the author, a correlation analysis was performed ($p < 0.05$) using the software IBM SPSS Statistics. We established a number of correlations between the text parameters and the author's gender (0 – woman, 1 – man). The following correlations ($p < 0.05$) are found: the number of content words / the number of function words (0.258); the number of nouns / the total of words (0.252), the number of function words / the number of nouns (0.297), (pronouns of all types + prepositions + pronominal adverbs) / the total of words (-0.269) and so forth.

The accuracy of the model assessed on test corpus is ~60%.

2. Second experiment. The study [8] is a follow-up of the search for the text parameters independent of its subject matter and consciously uncontrolled by the author and therefore impossible to imitate. As the analysis of scientific literature suggests, these are frequencies of sequences (bigrams) of parts of speech. The research using English-language materials has proved the analysis of the frequencies of different bigrams in texts to be efficient in authorship profiling [14].

96 texts were used for the study which were randomly selected from RusPersonality corpus. The frequencies of POS bigrams in each text (227 types of bigrams were overall identified) were calculated, bigrams were then selected which are found in no less than 75% of the analyzed texts.

The only bigram found to have a significant correlation with the gender of the author of the text is **prep_noun bigram**. Its Pearson's correlation coefficient is 0.215. Therefore, it can be stated that there is weak linear connection between the proportions of prep_noun bigrams in the text and the gender of its author, males typically score more on this parameter.

Selecting different types of linear functions revealed that this dependence is most accurately described by a four-parameter linear regression.

The model was tested on test set (texts not used for designing the model, 10 written by males, 10 written by females, mean length = 161 word). The model was found to be 65% accurate. It also should be noted that the model was considerably better at distinguishing females than males.

3. Third experiment. A number of the text parameters correlated with gender of their authors allowed us to design a regression models [8;9]. However, testing of the quality of the models showed that this type of approximation yields a low level of accuracy as the parameters of texts by individuals of different gender are usually in overlapping ranges. This makes it impossible to design a functional model as part of a multiparameter regression. Therefore, it was decided to design a few regression models instead. In order to design regression models, 1090 texts by 545 authors were randomly selected (two texts by each respondent) from *RusPersonality*.

The text parameters were only those that were not consciously controlled: indicators of lexical diversity of a text, proportions of parts of speech, and different correlations of parts of speech (a total of 78 parameters).

For each text parameter a regression model was designed based on an optimal selection considering the sign of a correlation coefficient and exclusion of statistical outliers. Let us show the suggested approach using an example of 5 text parameters correlated with the gender of an author ($p<0.05$): TTR (type-token ratio, $r = 0.390$), formality ($r = 0.315$), a proportion of prepositions and pronoun-like adjectives ($r = 0.243$), proportion of the 100 most frequent Russian words in a text ($r = -0.322$); a ratio of function words to content words in a text ($r= -0.295$).

In order to properly estimate the obtained result, let us determine the average arithmetic values from the solution of the five equations:

$$GENDER_1 = -0.669 + \left(2.622 TTR\right), \tag{2}$$

$$GENDER_2 = -0.637 + \left(0.971\ Formality\right), \tag{3}$$

$$GENDER_3 = -0.188 + \left(0.0432\ preposition + pronoun - like\ adjective\right), \tag{4}$$

$$GENDER_4 = 1.500 - \left(0.0303\ Frequent\right), \tag{5}$$

$$GENDER_5 = 1.392 - \left(0.0229\ Function\right) \tag{6}$$

In order to properly estimate the obtained result, let us determine the average arithmetic values from the solution of the five equations.

In order to estimate the suggested approach, we used a corpus of texts with contributions from 553 individuals (368 women, and 185 men, while two texts from each respondent were considered as one text). Their topic and length were identical to those used to design the regression models. Gender was correctly identified in 65% of women and 63% of men. Thus, the accuracy of the approach was 64%.

4 Classification models

For the current research we have chosen 556 respondents and each of them wrote two texts (a description of a picture and a letter to a friend). Each of two texts were joined and considered as one text with average length of 350 words. All the texts were split into the learning (80%), cross-validation (10%) and trial (10%) samples. We used different groups of features, in total 141 features:

1) *Emomarkers* – psycholinguistic markers of emotiveness based on morphological features [8];

2) *EmoDicts* – frequencies of emotional words (e.g., "Anxiety", "Discontent", the total of 37 categories [6]);

3) *Litvinova* – a set of parameters used in [9] which are ratios of PoS frequencies, number of sentences in a text, number of clauses, number of exclamation marks etc.

4) *PoS* – frequencies of part-of-speech (nouns, adjectives, adverbs, pronouns, numerals, particles, prepositions, verbs, conjunctions) [11];

5) *Sinto* – frequencies of different types of syntactic relationships between head and dependents [11].

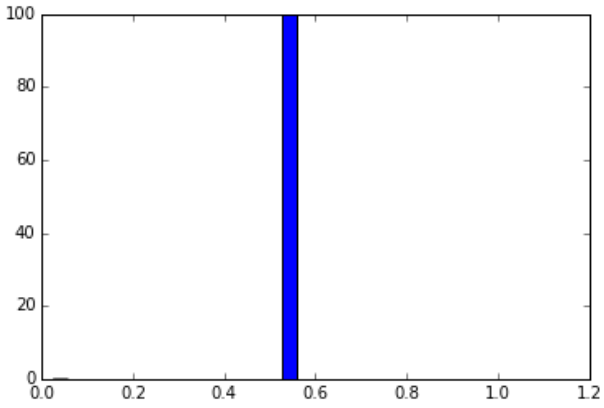
Comparative analysis of different machine learning algorithms has shown that ReLu is the most efficient classification algorithm with the F-score of 0.74 (see Table 1).

Table 1. F-scores for different classification algorithm

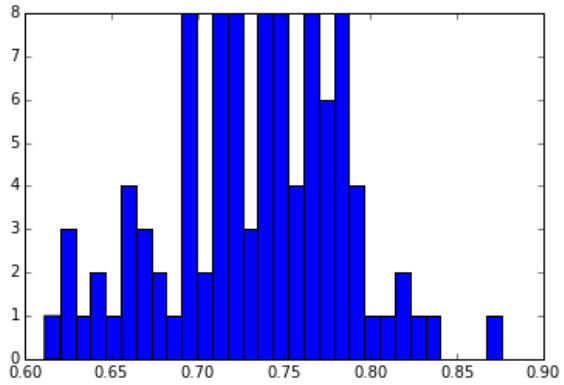
Model	Feature selection techniques	Mean F1-score (25 cycles)
Gradient Boosting	imp_quarter	0.72
adaBoosting	imp20	0.71
ExtraTrees	imp10	0.7
adaBoosting	common	0.7
Random Forest	imp10	0.7
PNN(sigma = 0.1)	imp10	0.68
SVM	PCA (30)	0.66
ReLU (1 Hidden Layer with 26 neurons)	imp10	0.74

In order to understand which groups of parameters yield the most accurate result, we designed graphs of f1-score distribution of the trained models (for SVM with a linear core) with different sets of parameters, each model was trained 100 times with a new combination of example sets. The selection was divided into the training (80%) and testing (20%) samples each time (see Fig. 1).

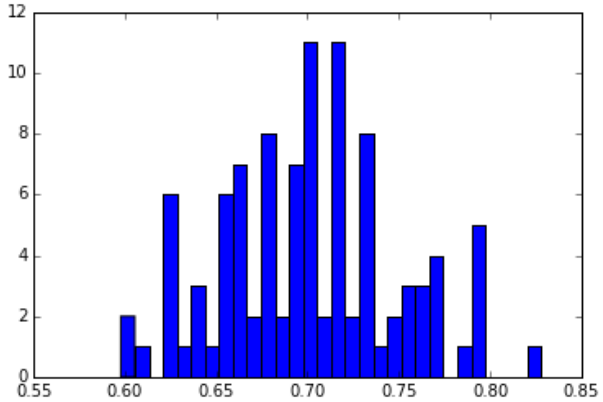
Fig. 1. F1-scores of the trained models (for SVM with a linear core) with different sets of parameters (the values of f1-score are plotted along the axis X, the number of models with a specified accuracy is plotted along the axis Y)



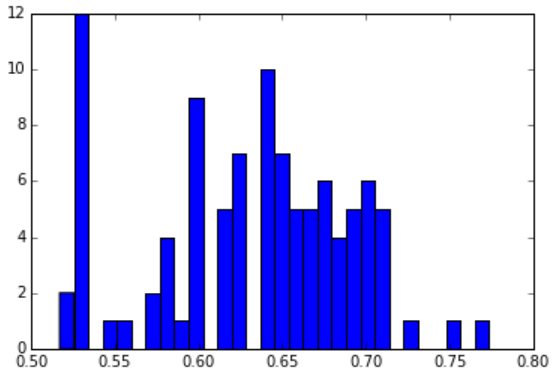
a) EmoMarkers



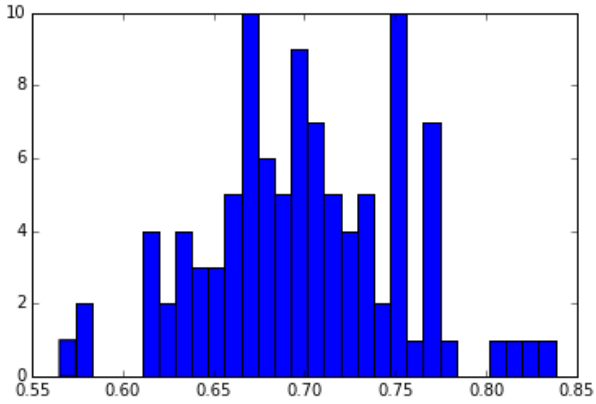
b) EmoDicts



c) Litvinova



d) PoS



e) Sinto

As Fig. 1 shows, the most informative parameters for identifying the gender of text author are different ratios of parts of speech, syntactic parameters, frequencies of various emotional words.

5 Conclusion and future work

The study performed for the first time for Russian-language texts confirms previously reported for English and some other languages findings that gender can be traced in texts beyond topic and genre using both regression and classification approach. It is shown that the author's gender is conveyed through specific syntactical and morphological patterns and use of emotion words. Comparative analysis of different machine learning algorithms has shown that ReLu is the most efficient classification algorithm with the F-score of 0.74. There are plans to expand the list of the text parameters and to test the obtained models on a Russian corpus of tweets and online chats.

Acknowledgements. This research is financially supported by the Russian Science Foundation, project No 16-18-10050 “Identifying the Gender and Age of Online Chatters Using Formal Parameters of their Texts”.

References

1. Burger, J. D., Henderson, J., Kim, G., Zarella, G.: Discriminating Gender on Twitter. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Edinburgh, 27-31 July 2011, pp. 1301-1309.
2. Cheng, Na, Chandramouli, R., Subbalakshmi, K.P.: Author gender identification from text. Digital Investigation 8(1), 78-88 (2011)
3. Ciot, M., Sonderegger, M, Ruths, D.: Gender Inference of Twitter Users in Non-English Contexts. In: Conference on Empirical Methods in Natural Language Processing (EMNLP) (2013).
4. Corney, M., Vel, O. de, Anderson, A., Mohay, G.: Gender-preferential text mining of e-mail discourse. In: Computer Security Applications Conference, 2002. Proceedings. 18th Annual, 2002, pp. 282-289.
5. Deitrick, W., Miller Z., Valyou B., Dickinson B., Munson T., Hu W.: Author Gender Prediction in an Email Stream Using Neural Networks. Journal of Intelligent Learning Systems and Applications 4(3) (2012)
6. Information Retrieval System "Emotions and feelings in lexicographical parameters: Dictionary emotive vocabulary of the Russian language.", Web: http://lexrus.ru/default.aspx?p=2876
7. Koppel, M., Argamon, S., Shimoni, A.R.: Automatically categorizing written texts by author gender. Literary Linguist. Comput. 17(4), 401–412 (2002)
8. Litvinova, T. A., Seredin, P. V. Litvinova, O. A.: Using Part-of-Speech Sequences Frequencies in a Text to Predict Author Personality: a Corpus Study. Indian Journal of Science and Technology 8(9) [S. l.], 93—97 (2015)
9. Litvinova, T. A.: Profiling the Author of a Written Text in Russian. Journal of Language and Literature 5(4): 210—216 (2014)
10. Rangel, F., Fabio, C., Rosso, P., Potthast, M., Stein, B., Daelemans W.: Overview of the 3rd Author Profiling Task at PAN 2015. In: Linda Cappellato and Nicola Ferro and Gareth Jones and Eric San Juan (eds.): CEUR Workshop Proceedings. Toulouse, France (2015) http://www.sensei-conversation.eu/wp-content/uploads/2015/09/15-pan@clef.pdf
11. Rybka, R., Sboev, A., Moloshnikov, I. Gudovskikh, D.: Morpho-syntactic parsing based on neural networks and corpus data. In: Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT), pp. 89 –95. IEEE, St. Petersburg (2015)
12. Sarawgi, R., Gajulapalli, K., Choi, Y.: Gender attribution: tracing stylometric evidence beyond topic and genre. In: EMNLP '11 Proceedings of the 15th Conference on Computational Natural Language Learning, pp. 78–86. Association for Computational Linguistics, Stroudsburg (2011)
13. Sboev, A., Gudovskikh, D., Rybka, R., Moloshnikov, I.: A Quantitative Method of Text Emotiveness Evaluation on Base of the Psycholinguistic Markers Founded on Morphological Features. Procedia Computer Science vol. 66, 307-316 (2015)
14. Wright, W. R., Chin D. N., Personality Profiling from Text: Introducing Part-of-Speech N-Grams, User Modeling, Adaptation, and Personalization. Lecture Notes in Computer Science, 8538:502-507 (2014).

Reproducing Russian NER Baseline Quality without Additional Data

Valentin Malykh[1,2], Alexey Ozerin[2]

[1] Institute for Systems Analysis of Russian Academy of Sciences,
9, pr. 60-letiya Oktyabrya, Moscow, 117312, Russia
http://www.isa.ru/
[2] Laboratory of Neural Systems and Deep learning,
Moscow Institute of Physics and Technology (State University),
9, Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia
http://www.mipt.ru/

Abstract. Baseline solutions for the named entity recognition task in Russian language were published a few years ago. These solutions rely heavily on the addition data, like databases, and different kinds of preprocessing. Here we demonstrate that it is possible to reproduce the quality of existing database-based solution by character-aware neural net trained on corpus itself only.

Keywords: named entity recognition, character awareness, neural nets, multitasking

1 Introduction

Named entity recognition is a well known task in natural language processing field. It is highly demanded in the industry and has a long history of academic research.

Current approaches are critically dependent on the size and quality of the knowledge-base used. The knowledge base should be kept up to date, which requires additional resources to be constantly involved.

In contrast our solution relies only on the text of the corpus itself without any additional data, except of the training corpus markup.

Contributions of the paper are following:

- We propose an architecture of artificial neural net as an alternative to the knowledge base based approach for the named entity recognition task.
- We provide results of the model tests on publicly available corpus for Russian language.

2 Related work

The first results for character-based named entity recognition in English language were presented in early 2000-s [1]. The close idea of character-based named entity

tagging was introduced in [2] for the Portuguese and Spanish languages, but our model does not use convolution inside. For the English language text classification (close task for the named entity recognition) character-aware architecture was described in [3], it is also basing on convolutions, so principally differs from our model. Previous research for Russian language hadn't been based on characters, but on words [4]. State of the art solution on the public corpus with named entity markup [5] is also word-level based.

One of the core ideas for our model comes from the character aware neural nets introduced recently in [6], [7]. Another idea, that of matching the sequences to train the artificial neural net to get the text structure is coming from [8]. Our solution is based on the multi-task learning which was introduced for natural language processing tasks in [9].

3 Model

The architecture of our recurrent neural network is inspired by [7]. The network consists of long short-term memory units, which were initially proposed in [10]. There are two main differences to the Yoon Kim setup [7]. First one is that our model predicts two things instead of one:

- the next character,
- a markup label for the current character.

Second one is that we do not use convolution, so we not exploiting word concept inside our architecture, only character concept. We suppose that model could learn the concept of word from data, and rely on this assumption while quality measurement. Prediction errors and gradients are calculated, and then weights are updated by truncated back-propagation through time [11].

3.1 Mathematical formulation

Let h_t be the state of the last neural net layer before softmax transformations (*hidden state*). The probability is predicted by standard sotfmax over the set of characters $\mathcal{C}$ and the set of markup labels $\mathcal{M}$:

$$Pr(c_{t+1}|c_{1:t}) = \frac{exp(h_t \cdot p_1^j + q_1^j)}{\sum_{j' \in \mathcal{C}} h_t \cdot p_1^{j'} + q_1^{j'}} \tag{1}$$

$$Pr(m_t|c_{1:t}) = \frac{exp(h_t \cdot p_2^i + q_2^i)}{\sum_{i' \in \mathcal{M}} h_t \cdot p_2^{i'} + q_2^{i'}} \tag{2}$$

Here p_1^j is j-th column in character output embedding matrix $P_1 \in \mathbb{R}^{k \times |\mathcal{C}|}$, q_1^j is a character bias term. p_2^i is i-th column in markup output embedding matrix $P_2 \in \mathbb{R}^{l \times |\mathcal{M}|}$ and q_2^i is markup bias term, k and l are character and markup embedding vector lengths.

The final negative log likelihood (NLL) is computed over the test corpus of length T:

$$NLL = -\sum_{t=1}^{T}(\log Pr(c_{t+1}|c_{1:t}) + \log Pr(m_t|c_{1:t})) \tag{3}$$

The diagram of our model could be found on the figure 1.

4 Experiments

The corpus parameters are presented at table 1, more details on it could be found in [5]. It can be obtained from the authors of the original paper by sending a request to `gareev-rm@yandex.ru` or to any other author of the original paper.

Table 1. Russian NER corpus statistics

Tokens	44326
Words & Numbers	35116
Characters	263968
Organization annotations	1317
Org. ann. characters	14172
Person annotations	486
Per. ann. characters	5978

Similar to [5] we calculate 5-fold cross-validation with precision (P), recall (R), and F-measure (F) metrics. The results of experiments are presented in table 2. Since we are working with characters we cannot use labelling produced for characters by our system directly, so we parse the produced markup for every token (which is known for us from the corpus) and take the label for the majority of characters in the token as a token label.

Table 2. 5-fold cross-validation of the NN-based NER.

Fold #	Person			Organization			Overall		
	P	R	F	P	R	F	P	R	F
1	93.09	93.32	93.20	68.75	78.57	73.33	63.25	71.94	67.32
2	94.85	94.16	94.51	64.29	73.90	68.76	59.38	67.86	63.33
3	90.91	93.37	92.12	66.22	65.52	65.87	58.45	58.76	58.60
4	90.45	91.74	91.09	68.02	77.48	72.45	60.12	68.56	64.06
5	94.03	93.06	93.54	62.15	68.81	65.31	57.06	61.40	59.15
mean	92.67	93.13	92.89	65.89	72.86	69.14	59.65	65.70	62.49
std	1.92	0.88	1.32	2.70	5.60	3.67	2.31	5.44	3.63

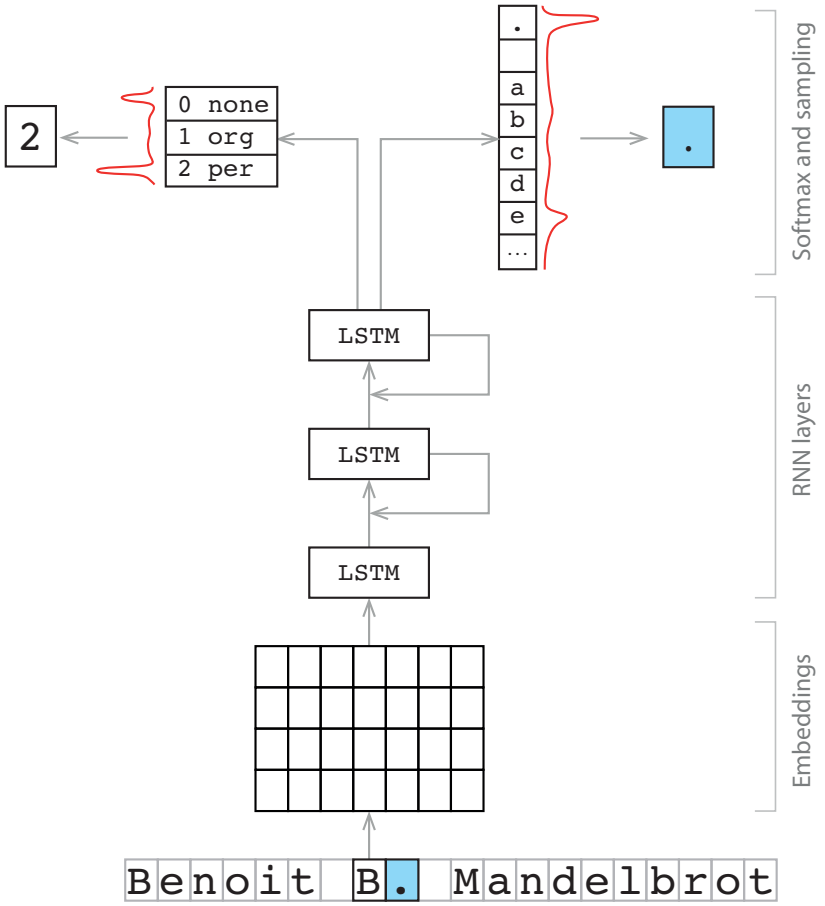


Fig. 1. Neural net architecture

5 Comparison

The results of comparison are presented on tables 3, 4, 5.

Table 3. Person class performance comparison.

System	Person					
	Precision		Recall		F-measure	
	mean	std	mean	std	mean	std
Best KB-based [5]	79.38	N/A	79.22	N/A	79.30	N/A
CRF-based [5]	90.94	4.04	79.52	2.91	84.84	3.33
NN-based	**92.67**	1.92	**93.13**	0.88	**92.89**	1.32

Table 4. Organization class performance comparison.

System	Organization					
	Precision		Recall		F-measure	
	mean	std	mean	std	mean	std
Best KB-based [5]	59.04	N/A	52.32	N/A	55.48	N/A
CRF-based [5]	**81.31**	7.44	63.88	6.54	**71.31**	5.38
NN-based	65.89	2.70	**72.86**	5.60	69.14	3.67

Table 5. Overall performance comparison.

System	Overall					
	Precision		Recall		F-measure	
	mean	std	mean	std	mean	std
Best KB-based [5]	65.01	N/A	59.57	N/A	62.17	N/A
CRF-based [5]	**84.10**	6.22	**67.98**	5.57	**75.05**	4.82
NN-based	59.65	2.31	65.70	5.44	62.49	3.63

On the person token class our system performed better than CRF-based one by all the metrics by the mean value and standard deviation. On the organisation class our system is better by recall and comparable by F-measure with CRF-model. In overall case our system was on par with knowledge-base approach performance in F-measure and in recall with CRF-model.

6 Conclusion

We applied character aware RNN model with LSTM units to the problem of the named entity recognition in Russian language. Even without any preprocessing

and supplementary data from external knowledge-base the model was able to learn solution end-to-end from the corpus with markup. Results demonstrated by our approach are on the level of existing state of the art in the field.

The main weakness of proposed model is differentiation between person and organization tokens. This is due to the small size of the corpus. A possible solution is pre-training on a large corpus such as Wikipedia, without any markup, just to train internal distributed representation of a language model. We presume that such pre-training would allow RNN to beat CRF-model.

Another direction of our future work is addition of attention as it was demonstrated to improve performance on character-level sequence tasks [12].

References

1. Klein, D., Smarr, J., Nguyen, H., Manning, C.D.: Named entity recognition with character-level models. In: Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4, Association for Computational Linguistics (2003) 180–183
2. dos Santos, C., Guimaraes, V., Niterói, R., de Janeiro, R.: Boosting named entity recognition with neural character embeddings. In: Proceedings of NEWS 2015 The Fifth Named Entities Workshop. (2015) 25
3. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Advances in Neural Information Processing Systems. (2015) 649–657
4. Popov, B., Kirilov, A., Maynard, D., Manov, D.: Creation of reusable components and language resources for named entity recognition in russian. In: Conference on Language Resources and Evaluation. (2004)
5. Gareev, R., Tkachenko, M., Solovyev, V., Simanovsky, A., Ivanov, V.: Introducing baselines for russian named entity recognition. In: Computational Linguistics and Intelligent Text Processing. Springer (2013) 329–342
6. Bojanowski, P., Joulin, A., Mikolov, T.: Alternative structures for character-level rnns. arXiv preprint arXiv:1511.06303 (2015)
7. Kim, Y., Jernite, Y., Sontag, D., Rush, A.M.: Character-aware neural language models. arXiv preprint arXiv:1508.06615 (2015)
8. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in neural information processing systems. (2014) 3104–3112
9. Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th international conference on Machine learning, ACM (2008) 160–167
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8) (1997) 1735–1780
11. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)
12. Golub, D., He, X.: Character-level question answering with attention. arXiv preprint arXiv:1604.00727 (2016)

A Static Code Search Technique to Identify Dead Fields by Analyzing Usage of Setup Fields and Field Dependency in Test Code

Abdus Satter, Amit Seal Ami, and Kazi Sakib

Institute of Information Technology, University of Dhaka
Dhaka 1000, Bangladesh
{bit0401,amit.seal,sakib}@iit.du.ac.bd

Abstract. Dead field is one of the most common test smells found in the test code which is responsible for degrading performance and creating misapprehension about the code. The reason of its occurrences is that in most of the cases, developers initialize setup fields without considering the usage of those fields in the test methods. In this paper, an automatic dead field identification technique is proposed where the test code is statically searched for identifying the usage of all the setup fields. It does so by figuring out all the setup fields which are initialized in the setup method or its invoked methods. After that, it detects such fields which are used by at least one test method directly or indirectly. In addition, field dependency is resolved to find all the fields on which used setup fields depend. At last, all the unused setup fields are gathered and considered as dead fields. To evaluate the technique, it was implemented in the form of a tool and two open source projects were run on it. It has been seen that it identifies all the dead fields in those projects correctly and performs better than existing dead field detection techniques.

Keywords: test smell, dead field, setup fields, test fixture, test smells, test code comprehension, code search

1 Introduction

Dead fields are the initialized fields in the setup method of a test class that are never used by any test method. However, manually scrutinizing test code to find dead fields slows down the production of software, and may induce bugs while applying refactoring methods to stump out this smell. On the other hand, when test code fails to convey its intent is considered to have the test smells which have no impact on the behavior of the test code but causes the attrition of the test code quality. Dead field is one of those which degrades the performance of the test code as well as maintainability by unnecessarily using computational resources and creating misunderstanding among the developers. So, dead fields should be identified and removed from the test code to maintain the quality. Dead fields can be identified by analyzing the test fixtures and test methods in the test code. However, the major challenges are to find all setup fields in the

test fixtures, resolve field dependency among those and figure out the usage of those fields in test methods automatically.

Generally, a test fixture defines the configuration of the system under test including setup methods. Those setup methods are invoked before execution of any test case to ensure that all the setup fields are initialized properly for running the test cases. On the other hand, after preparing the system for testing, test methods use their required setup fields directly or indirectly through invoking other method(s). To identify dead fields, unused setup fields are required to be detected for which all the setup fields and their usage in the test methods are needed to be analyzed. However, in order to identify setup fields, all the setup methods in the test fixture are required to be analyzed. In addition, to find the usage of the setup fields, dependency relationship among the setup fields and those fields' usage in test methods need to be resolved.

Martin Fowler coined code smell [1] and later, van Deursen first introduced the concept of test smells in test code [2]. Michael Grielar and van Deursen identified five new test smells including dead fields and developed a tool named TestHound[1] to identify those smells by analyzing test fixture [3]. The tool performs well in identifying those test smells but for dead field detection, manual code inspection is required to resolve field dependency and usage of setup fields among the test methods. TestLint, another automatic test smells identification tool, can deal with some test smells by finding the properties of those smells in the test code [4]. However, this tool cannot handle dead fields in the test code through test fixture and test method analysis because dead fields have not been considered here. Although Bart van Rompaey proposed a metrics-based approach based on the unit test concept to identify eager test smell, the author did not address any metric to automatically detect dead fields [5, 6]. Bavota disclosed the distribution and impact of test smells in software maintenance but no approach was explained to automatically identify dead fields in his analysis [7].

In this research, a technique named Dead Field Identifier (DFI) is proposed to identify dead fields automatically by analyzing usage of setup fields and field dependency in test methods. Initially, all the fields in the test code are searched and gathered from test code. As header fields[2] are also considered as setup fields, so all the header fields are figured out from the identified fields by parsing the code. Later, setup method and all other methods invoked directly or indirectly by it are identified. The body of those methods are extracted to find all the setup fields in the code. To find usage of those fields, all the test methods and other methods invoked by those are obtained and fields which are used in those methods are detected. Usually, it is found in the code that a setup field which is used in at least one test method may depend on one or more other setup fields which are never used by any test method. So, those fields are identified through analyzing field dependency among the setup fields and considered as used setup fields. At the end, all unused fields are separated from the setup field list and those are marked as dead fields.

[1] http://www.swerl.tudelft.nl/twiki/pub/MichaelaGreiler/TestHound/TestHound

[2] Header fields are those fields which are initialized in the class header

In order to assess the technique, a tool is implemented based on it. The proposed technique requires test code related information like fields in the test class, method signature, method body etc. For this reason, test code is converted into compiler centric Abstract Syntax Tree (AST) by the tool where AST is a semi-structured form of the code [8, 9]. This tree based representation assists to find required information more easily for dead field detection than searching in the raw test code [10]. Two open source projects (eGit[3] and EquationSolverTest[4]) are used for the justification of the technique. Both projects are run on DFI and TestHound for comparative analysis of the technique. Manual inspection is also carried out to ensure the correctness of the result. While analyzing the results, it is seen that for eGit, 3 percent of the fields could not be identified as used fields whereas DFI figures out all the dead fields by properly identifying all used setup fields. On the other hand, 82 percent setup fields can not be detected for EuqationSolverTest and as a result 67 percent setup fields are not considered as dead fields by it. However, DFI resolves field dependency among setup fields and finds the usage of those in test methods correctly. For this reason, it performs better than TestHound by identifying all the setup fields and dead fields correctly in the project.

The rest of the paper is organized as follows. Section 2 describes related works in detecting dead fields. The proposed technique is discussed in Section 3. Section 4 presents implementation and result analysis of the proposed technique. Conclusion is drawn in Section 5.

2 Related Work

The presence of dead field in the test code indicates incomplete or deprecated software development activities. This smell is a recent contribution in the literature. Several researches have been carried out so far for analyzing the impact of test smells in the test code. Besides, researchers proposed different techniques to identify and remove those smells from the code. These are outlined as follows.

van Deursen et al. first described the concept of test smells [11, 12]. They identified a list of eleven different test smells such as Mystery Guest, Resource Optimism, Test Run War, General Fixture, Eager Test, Lazy Test, Assertion Roulette, Indirect Testing, For Testers Only, Sensitive Equality, and Test Code Duplication. They discussed about the characteristics of the smells and appropriate refactoring mechanism to remove those, but they did not provide any technique for automatically identifying dead field in the test code because this smell was not discovered at that time.

A metrics-based approach was proposed by Bart van Rompaey et al. [5, 13] for the detection of two test smells which were test fixture and eager test to increase the quality of test cases. To identify test fixture, they used several metrics like setup size, fixture size, and fixture usage. Setup size is the combination of the number of method or attribute references to non-test object from the setup

[3] http://www.eclipse.org/egit/

[4] https://github.com/rifatbit0401/EquationSolverTest

method of a test case, and number of production type used in the test code. They also defined fixture size as number of fixture elements and production type in the fixture. For eager test identification, they used production type method invocation as metric which is the number of invocations to the methods in the production code from a test command. Their result was compared against manual inspection. The technique worked well in identifying test fixture and eager test smell. However, the metrics that were used to identify those smells are not adequate enough to detect dead field in the test code as its characteristics are different.

In order to understand the distribution of unit test smells and the impact of those smells on software maintenance, Gabriele Bavota et al. conducted an empirical analysis regarding this [7]. Two studies were carried out for the analysis where one was an exploratory study and another was a controlled experiment. The exploratory study was performed for the analysis of the distribution of test smells. On the other hand, the controlled experiment was carried out for analyzing the impact of test smells on the comprehension of test code during software maintenance. Although they provided an insight about the distribution and impact of test smells while managing test code, they did not provide any approach to automatically detect dead fields in the code. The reason is that they only analyzed the impact and distribution rather than detection of test smells.

Stefan Reichhart et al. developed a tool named TestLint for assessing the quality of test code [4]. This rule-based tool identifies static test smells such as Guarded Test, OverReferencing, Assertionless Test, Long Test, Overcommented Test, and so on. It performs so by parsing the source code, analyzing the source tree, detecting patterns, and computing metrics on the test code. All the rules used to develop the tool were the characteristics of those smells [2, 14, 15]. However, the tool can not identify dead field in the test code because no metric was defined for the identification of this smell.

Manuel Breugelmans and Bart van Rompaey presented a tool called TestQ for exploring structural and maintenance characteristics of unit test suites [6]. It allows developers to visually explore test suites and quantify test smelliness. The tool could identify twelve different test smells proposed by van Deursen [5]. For the detection, the tool uses a list of metrics defined by the authors such as number of invoked framework asserts for Assertionless, number of invoked descriptionless asserts for AssertionRoulette, number of invoked production methods for EagerTest, and so on. However, the tool can not detect dead field in the test code because they did not define any metric or strategy for it.

A static analysis technique to identify test fixture related smells in the test code was presented by Michaela Greiler et al. [3]. Here they introduced five new test smells which are Test Maverick, Dead Fields, Lack of Cohesion of Test Methods, Obscure In-Line Setup, and Vague Header Setup. To identify those smells, they developed a tool named TestHound. It takes the test code, all dependencies and all test cases as input. Next, it analyzes the code, finds the smells, and provides a report describing all identified test smells in the code. The tool was assessed by running on three projects (eGit, HealthCare and Mylyn). However,

it produced false positive results while detecting dead fields due to not being able to resolve field dependency and find usage of setup fields in the test code. So, manual inspection was performed to identify dead fields correctly.

Although dead field is a recently introduced test smell in the literature, some significant works have been performed in identification of test smells so far. Researchers explained the impact of test smells in test code maintenance and proposed different techniques to detect test smells like metrics based approach, rule based assessment, test fixture analysis and so on. Some of those could identify dead fields in the test code but the outcome is not accurate enough. Sometimes it is seen that those techniques provides false positive result which ultimately induces serious impact while managing the code. For that reason, test code is needed to be inspected manually for making sure the correctness of the result in dead field detection. So, automatically identifying dead fields in the code properly is still an open problem in the literature.

3 The Proposed Technique

The intent of this research is to develop a technique named Dead Field Identifier (DFI) to identify dead fields in the test code for making the code more maintainable and comprehensible by removing those fields. For the identification, firstly, it is required to identify all the invoked methods for any method in the test code. In addition, all the setup fields are required to be obtained and usage of those fields are needed to be identified in the code which assist to detect dead fields. So the technique for the identification comprises several steps like invoked method identification, setup field detection, finding usage of setup fields and dead field identification which are described in the following subsections.

Algorithm 1 Invoked Method Identification

Require: A method (M) for which all the methods invoked directly or indirectly by it will be identified and an empty list L to store invoked methods

```
1: procedure GetAllInvokedMethod(M)
2:     if M ∉ L then
3:         add M into L
4:     end if
5:     initialize an empty list N to store methods invoked by M
6:     get all methods invoked by M through parsing its body
7:     store those methods into the list N
8:     for each m ∈ N do
9:         A ← GetAllInvokedMethod(m)
10:        Insert all items in A into L
11:    end for
12:    return L
13: end procedure
```

3.1 Invoked Method Identification

In order to identify invoked method(s) in the test code Algorithm 1 is developed. Usually, the first step to identify dead fields in the test code is to identify all the methods invoked directly or indirectly by any method in the code. This is required because fields in the test class may be initialized by any method invoked directly or indirectly by the setup methods. Even setup field(s) may not be used directly by a test method but may be used by other methods which are invoked by the test method directly or indirectly.

In Algorithm 1, the procedure $GetAllInvokedMethod$ takes a method as input and returns a list of all methods invoked directly or indirectly by the method. For this, first of all, a list is initialized to store all invoked methods and the body of the inputted method is parsed to identify all the methods invoked by it which are inserted into another list (Algorithm 1 Line 5-7). A loop is used to identify all the invoked methods for each method in the list by recursively calling $GetAllInvokedMethod$. For each iteration, the corresponding method is also added into the list which is responsible for containing all invoked methods (Algorithm 1 Line 8-11).

3.2 Finding Setup Fields

Setup fields in the test code are those which are initialized in the implicit setup procedures or the class header. All the setup fields in the test code are required to be identified because such setup fields are considered as dead fields which have never been used by any test method in the test code.

Algorithm 2 describes a procedure $GetAllSetUpFields$ which works on given test code and provides a list of all setup fields in the code. Initially two lists are initialized - one is to store all setup fields and another is to store all the fields by parsing the test code (Algorithm 2 Line 2-3). In the loop, all the header fields are identified from the list of fields and those are added to the setup field list as header field is also considered as setup field (Algorithm 2 Line 4-8). After that, from rest of the fields those which are initialized in the implicit setup are added to the list of setup fields (Algorithm 2 Line 9-22).

3.3 Finding Usage of Setup Fields

After identifying all setup fields following the previous step, the usage of all these fields are required to be found in the test code. This will help to detect which setup fields are never been used by any test method in the test code.

In Algorithm 3, all the test methods and all the setup fields in the test code are identified and stored in two different lists respectively (Algorithm 3 Line 3-5). For each identified test method, the procedure $GetAllInvokedMethod$ is called to obtain all the methods invoked directly and indirectly by the method (Algorithm 3 Line 6-8). After that, the body of each invoked method and the test method are checked to identify which setup fields are used in the body and such fields are added to the used setup field list (Algorithm 3 Line 9-16). At last, the

Algorithm 2 Finding Setup Fields

Require: Test code T for identifying all setup fields in T

procedure GetAllSetUpFields(T)
  initialize an empty list S to store setup fields
  identify all the fields in T using parser and store those fields in the list F
  for each $f \in F$ **do**
    if f is header field **then**
      Add f to S
    end if
  end for
  find setup method M by parsing T
  create an empty list I to store method
  $I \leftarrow$ GetAllInvokedMethod(M)
  add M to I
  for each $m \in I$ **do**
    for each $f \in F$ **do**
      if $f \in S$ **then**
        continue
      end if
      if f is initialized in m **then**
        add f to S
      end if
    end for
  end for
  return S
end procedure

Algorithm 3 Finding Usage of Setup Fields

Require: Test code T for finding usage of setup fields in the test code

procedure GetAllUsedSetUpField(T)
  initialize an empty list U to store all used setup fields in T
  initialize an empty list M to store all test methods in T
  identify all test methods by parsing T and add those into M
  $S \leftarrow$ GetAllSetUpFields(T)
  for each $m \in M$ **do**
    $L \leftarrow$ GetAllInvokedMethod(m)
    add m to L
    for each $i \in L$ **do**
      Get the body of the method (i) and save it in b
      for each $f \in S$ **do**
        if f is used in b and $f \notin U$ **then**
          add f to U
        end if
      end for
    end for
  end for
  return U
end procedure

Algorithm 4 Dead Fields Detection

```
Require: Test code T to identify dead fields in the code
 1: procedure GetAllDeadField(T)
 2:     S ← GetAllSetUpFields(T)
 3:     U ← GetAllUsedSetUpField(T)
 4:     F ← S − U
 5:     initialize a list D to store dead fields
 6:     for each f ∈ F do
 7:         flag ← false
 8:         for each i ∈ U do
 9:             if i depends on f for initialization in the implicit setup then
10:                 flag ← true
11:             end if
12:         end for
13:         if flag = false then
14:             add f to D
15:         end if
16:     end for
17:     return D
18: end procedure
```

list of all used setup fields are returned by the procedure $GetAllUsedSetUpField$ (Algorithm 3 Line 18).

3.4 Dead Fields Detection

Subsection 3.3 provides all the setup fields that are used by at least one test method directly or indirectly. However, such setup fields can be found in the test code, which are not being used by any test method, but some used setup fields may depend on those fields for initialization. So, those fields are not considered as dead fields. For finding all those fields, incorporating those with the list of fields obtained using subsection 3.3 and finally providing a list of all identified dead fields in the test code, Algorithm 4 is used for implementation.

To detect dead fields, all the setup fields and used setup fields are gathered (Algorithm 4 Line 2-3). A list is used to store all the setup fields which are not used by any test method (Algorithm 4 Line 4). The nested loops identify which setup fields of the list are never used for the initialization of any used setup field (Algorithm 4 Line 6-16). All those unused fields are considered as dead fields which are returned by the procedure $GetAllDeadField$ as a list (Algorithm 4 Line 17).

Complexity Analysis

The overall complexities of $GetAllInvokedMethod$, $GetAllSetUpFields$, $GetAllUsedSetUpField$, and $GetAllDeadField$ are $O(p)$, $O(pq)$, $O(prs)$, and $O(pq + prs + mn)$ respectively. Here, p, q, r, s, m and n are number of invoked

methods, number of fields, number of test methods, number of setup fields, number of unused setup fields, and number of used setup fields correspondingly.

4 Implementation and Result Analysis

In order to evaluate DFI, a tool is implemented based on it. TestHound [3] is used for comparative analysis with DFI. At last, manual inspection is carried out to make sure the correctness of the result which is provided by DFI.

4.1 Environmental Setup

This subsection outlines the software tools required for the experimental analysis. For this analysis, DFI is developed using Java programming language. Although the tool works to identify dead fields in the test code written using Java, the approach proposed here is platform independent and only the facts extraction aspect is language specific. So, the technique can easily be implemented in any programming language. Some other tools are also used in the experiment and those are addressed as follows.

- Eclipse Juno[5]: Java IDE for the development of DFI
- Byte parser[6]: Java byte code parser which has been developed to parse java byte code and construct AST
- Maven[7]: Apache build manager for building the java projects used as the dataset in the experiment

Table 1. Experimented Projects

Project Name	Line of Code	Number of Test Class
EquationSolverTest	800	4
eGit	130k	87

For the analysis, two open source projects have been used which are depicted in Table 1. One of those is EquationSolverTest which is developed to solve equation having different expressions. It is an open source project and it has 800 lines of code as well as 4 test classes. Another is eGit which is also an open source Eclipse integrated version control system. It consists of 130K lines of code and 87 test classes. Both are available in the GitHub.

4.2 Comparative Analysis

As we have said above, two different sized projects are used for the experiment to observe the behavior of the proposed technique. One of those is eGit which is

[5] https://eclipse.org/juno/
[6] https://github.com/rifatbit0401/ByteParser
[7] https://maven.apache.org/

Table 2. Result for EquationSolverTest

Class Name	No. Test Method	No. of Setup Fields			No. of Dead Fields		
		TestHound	DFI	Manual Inspection	TestHound	DFI	Manual Inspection
Simulate EquationTest	4	0	5	5	0	4	4
Expression FormatterTest	3	1	4	4	1	3	3
Expression Simulation ResultTest	4	2	7	7	2	2	2
OperationTest	4	0	1	1	0	0	0

large in size, and another is EquationSolverTest which is comparatively small. The results obtained using those projects are explained as follows.

Result Analysis for EquationSolverTest: For comparative analysis, initially the project EquationSolverTest is run by TestHound and DFI. In addition, manual inspection is also performed on the code. Table 2 summarizes the result produced by the tools and manual inspection. In the table, it is seen that there are four test classes. Comparative analysis for those classes are described below.

For the test class SimulateEquationTest, TestHound can not identify any setup field whereas DFI detects 5 setup fields as well as 4 dead fields from those. The outcome of DFI is equal to the result of manual inspection. The reason is that TestHound can not identify those setup fields which are initialized in the methods invoked by setup method, but DFI considers all those methods and checks the initialization of setup fields.

In the test class ExpressionFormatterTest, there are 4 setup fields where one is header field and others are initialized through indirect method invocation by the setup method. TestHound detects no usage of the header field and thus, considers it as dead field, but others are ignored because of the same reason as stated earlier. However, DFI identifies all those and recognizes as dead fields.

Both tools identify two dead fields correctly for the test class ExpressionSimulationResultTest. However, TestHound identifies 2 setup fields out of 7 because those two are header fields and rest 5 are initialized in the setup method which are not considered in it. On the other hand, DFI checks the setup method as well as header field, that is why it detects all setup fields.

There is a single header field in test class OperationTest and this field is used in all 4 test cases. As both tools can detect header fields and usage of setup fields in test cases so those tools provide the same result for the test class.

Result Analysis for eGit: DFI is also run on a module of eGit named *org.eclipse.egit.core.test*. There are 13 test classes and 46 test methods in total. The result provided by the tool for the project is shown in Table 3. According to the table, DFI identifies 78 setup fields and 6 dead fields. To ensure the cor-

rectness of the result, manual inspection is performed and the same outcome is produced. During manual inspection, it is found that test classes which extend the same super class contain dead fields. This is because setup fields are initialized in the super class but never been used by any test method in the subclasses. However, DFI first identifies all the fields of a test class. Later, all the inherited fields are accumulated with those if the class extends another class. After that, it identifies setup fields among those by analyzing all methods' body invoked by the setup method and detects dead fields by finding usage of those fields in test methods. For this reason, DFI's result is the same to the manual inspection's outcome. However, 3% of the fields in this project could not be identified as field usage by TestHound [3]. So, in comparison with it, DFI performs better than it in dead field identification.

Table 3. Result of eGit by DFI

	DFI	**Manual Inspection**
Number of Test Class	13	13
Number of Test Method	46	46
Number of Setup Field	78	78
Number of Dead Field	6	6

DFI and TestHound, both can identify dead fields in the test code. However, TestHound can not detect dead fields correctly due to not handling some cases properly like setup fields initialization in a method invoked by setup method, field dependency among setup fields, and usage of setup fields by test methods indirectly. On the other hand, DFI can appropriately deal with those and as a result it detects dead fields correctly in the test code.

5 Conclusion

The presence of dead fields in the test code reduces the manageability and comprehensibility of the code. To detect those fields in the code, an automatic identification technique is introduced. A tool is also implemented based on the technique which identifies dead fields in the test code.

The technique first identifies all the fields in the test code through static code search. Setup fields are identified from those by analyzing the initialization of those fields in the setup method and its invoked method. At last, usage of those fields in test methods and dependency relationship among those fields are resolved to figure out dead fields in the code.

For the experimental analysis of the approach, two open source projects were run on it. The result of DFI was compared to another tool named TestHound. The experimental result shows that DFI identifies all the dead fields in those projects correctly and performs better than TestHound. In future, more open source projects and industrial projects will used to evaluate the technique.

Acknowledgment This work is supported by the University Grants Commission, Bangladesh under the Dhaka University Teachers Research Grant No-Regi/Admn-3/2016/46897. The authors would like to thank Sheikh Muhammad Sarwar for his contribution in the revision phase of the paper.

References

1. Martin Fowler. *Refactoring: improving the design of existing code*. Pearson Education India, 1999.
2. Arie van Deursen, Leon Moonen, Alex van den Bergh, and Gerard Kok. *Refactoring test code*. CWI, 2001.
3. Michaela Greiler, Arie van Deursen, and Margaret-Anne Storey. Automated detection of test fixture strategies and smells. In *Proceedings of the Sixth International Conference on Software Testing, Verification and Validation (ICST), 2013 IEEE*, pages 322–331. IEEE, 2013.
4. Stefan Reichhart, Tudor Gîrba, and Stéphane Ducasse. Rule-based assessment of test quality. *Journal of Object Technology*, 6(9):231–251, 2007.
5. Bart Van Rompaey, Bert Du Bois, Serge Demeyer, and Matthias Rieger. On the detection of test smells: A metrics-based approach for general fixture and eager test. *Software Engineering, IEEE Transactions on*, 33(12):800–817, 2007.
6. Manuel Breugelmans and Bart Van Rompaey. Testq: Exploring structural and maintenance characteristics of unit test suites. In *WASDeTT-1: 1st International Workshop on Advanced Software Development Tools and Techniques*, 2008.
7. Gabriele Bavota, Abdallah Qusef, Rocco Oliveto, Andrea De Lucia, and David Binkley. An empirical analysis of the distribution of unit test smells and their impact on software maintenance. In *Proceedings of the 28th IEEE International Conference on Software Maintenance (ICSM), 2012*, pages 56–65. IEEE, 2012.
8. David E Langworthy, John L Hamby, Bradford H Lovering, and Donald F Box. Tree-based directed graph programming structures for a declarative programming language, October 23 2012. US Patent 8,296,744.
9. Peter Buneman. Semistructured data. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 117–121. ACM, 1997.
10. Terence Parr. *The definitive ANTLR 4 reference*. Pragmatic Bookshelf, 2013.
11. A. van Deursen, L. Moonen, A. van den Bergh, and G. Kok. Refactoring test code. In *Proceedings of the 2nd International Conference on Extreme Programming and Flexible Processes (XP2001)*, pages 92–95. University of Cagliari, 2001.
12. A. van Deursen, L. Moonen, A. van den Bergh, and G. Kok. Refactoring test code. In G. Succi, M. Marchesi, D. Wells, and L. Williams, editors, *Extreme Programming Perspectives*, pages 141–152. Addison-Wesley, 2002.
13. Bart Van Rompaey, Bert Du Bois, and Serge Demeyer. Characterizing the relative significance of a test smell. In *Proceedings of the 22nd IEEE International Conference on Software Maintenance, 2006. ICSM'06.*, pages 391–400. IEEE, 2006.
14. Gerard Meszaros. *xUnit test patterns: Refactoring test code*. Pearson Education, 2007.
15. Gerard Meszaros, Shaun M Smith, and Jennitta Andrea. The test automation manifesto. In *Extreme Programming and Agile Methods-XP/Agile Universe 2003*, pages 73–81. Springer, 2003.

Verb-Noun Collocation and Government Model Extraction from Large Corpora

Vladislav Tushkanov, Oksana Dereza

National Research University Higher School of Economics,
v.tushkanov@outlook.com, oksana.dereza@gmail.com

Abstract. Knowing the government model, or argument structure, of a verb is crucial for many NLP tasks. In this article, a method of automatic extraction of verbs from large annotated corpora is devised. This method allows to computationally efficiently extract government models and particular arguments for every verb using a simple window-based approach by iterating through each sentence with a window of fixed size and applying frequency filters to filter out noise.

Keywords: collocations, verb government models, argument structure, corpus methods, parsing

Topic Modeling without Generative Probabilistic Model: An Approach and its Validation *

Mikhail G. Kreines, Elena M. Kreines

BaseTech LLC, Raketny boulevard, 11-7-164, Moscow, 129366, Russia
mkrf@yandex.ru

Abstract. We propose a novel approach to compute a model of semantic structure of natural language texts corpora based on the models of the texts. This approach differs from LSI and standard ways of topic modeling (D. Blei, 2012). We do not use matrix factorization and probabilistic distributions of the words given a priory. As the base for the corpora model construction we use the set of the original vector models of the corpora texts. The ways of empirical validation and applications of the models are considered.

Keywords: natural languages, texts, text collections, models, Information Retrieval

* The work is partially financially supported by the Ministry of Education and Science, Russian Federation (Contract N. 14.579.21.0090 from 27.11.2014, project identifier RFMEFI57914X0090).

Evaluation of Ontology Quality based on Analysis of Relations in Concept Lattices

Bato Merdygeev[1], Sesegma Dambaeva[1]

[1]East Siberia State University of Technology and Management, Ulan-Ude, Russia
mainisjusticeone@gmail.com
damseg@gmail.com

Abstract. The paper presents an approach to evaluation of the quality of domain ontology. The approach is based on construction of concept lattice based on ontology relations. The approach allows to evaluate the completeness of the ontology relations. The result of this analysis helps to draw conclusions about the overall quality of the ontology.

Keywords: ontology, domain, ontology analysis, concept lattice, relation, evaluating, completeness of the ontology relations

1 Introduction

In computer science, the term "ontology" means the formal representation of knowledge. It is used as a form of knowledge representation of the real world, or part of it [1].

Currently, many intelligent systems use ontology as a knowledge base. The effectiveness of this system depends on the effectiveness of knowledge represented in the ontology. Regardless of the type of ontology its creation is a laborious and expensive task. At the same time there is a possibility to receive of ineffective product as a result in accordance with the obsolescence of developed knowledge, priorities change with time or just give incorrect or contradictory knowledge a part of the ontology. To avoid it is necessary to evaluate the quality of ontology at every stage of its production. In the existing ontology analysis methods are based on the expert evaluation. Experts in this case often act domain experts or knowledge engineers. The main problem here is the amount of time required for checking the quality of the ontology. Modern methods provide a variety of tools for ontology analysis, but most of them are only effective in ontologies with a certain structure. Therefore, a search for new approaches to the analysis of the quality of ontology of various structures is needed.

One such approach could be the approach to ontology evaluation, based on an analysis of the relations between the terms of concept lattice. This approach analyzes the various inconsistencies that are detected by comparing the basic structure of the relations of ontology and concept lattices constructed on the basis of the same relations. Thus, the approach makes it possible to calculate the completeness of the ontology relations.

We introduce some definitions of key terms used in paper on the basis of [10].

The term is a sign of a special semiotic system, which is the minimum carrier of scientific knowledge, and it is the short name of an established concept of having a definition.

Concept is knowledge, which is expressed by this term at the conceptual modeling domain.

According to [1] conceptual objects are divided as follows:

- entities (tangible and intangible objects);
- properties (quantitative, qualitative, relative);
- actions (operations, processes, state);
- dimensions (time, position, space).

Conceptual relations are divided as follows:

- quantitative relations (relations of identity, inclusion, exclusion, intersection, union);
- qualitative relations (hierarchical and functional relations).

2 Domain ontology

In [2] the history of the sign in semiotics and logic was analyzed. Categories and their design of signs representing them were defined on the basis of a pentagon of Nikitina S.E., described in [11], and the concept structure and classification of conceptual objects of Dahlberg.

This approach of building structures of signs of conceptual objects of Dahlberg as the main categories of abstraction allows you to create a common conceptualization of the domain, which will be able to understand the different systems.

On the basis of this approach the basic design of structure of the terms of the domain ontology, proposed in [2], was created. Therefore the object of the analysis the studied approach is the ontology built on the basis of this ontology. The ontology is represented in this case in the form of groups of related terms, divided into categories: concept, action, state, event, property, quantity. Each term is a vector structure, a certain term category. The structure contains a full description of the term, including its name, relations to other terms meta-signed representation, etc.

Each term in this case is represented by a specific set of names, definitions and relations. **Categorical sign** is a sign that represents the general structure of the term of a certain category. Construction of categorical sign is represented as a vector of sets of definitions and relations represented by the term. Each categorical sign corresponds to one category of ontology terms.

This representation of domain ontology introduces sematic differences between terms of different categories, making this ontology structure semantically active.

Consider some of the categorical construction signs and the possible relations between the ontology terms.

"Concept" and "Action" categories have most semantic meaning in the ontological structure than others.

The design "Concept" sign is eight:

$$Concept = < t, D, P, A, C, S, T, M > \quad (1)$$

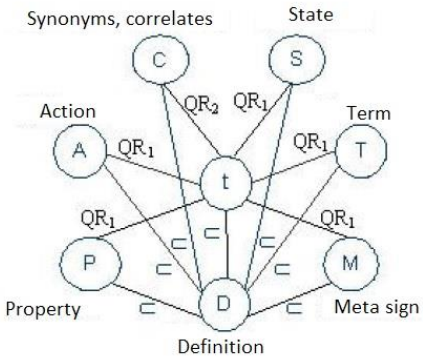


Fig. 1. Graphic representation of "concept" sign

The design of the "Action" sign is nine:

$$Action = < a, D, P, SO, C, I, A, E, M > \quad (2)$$

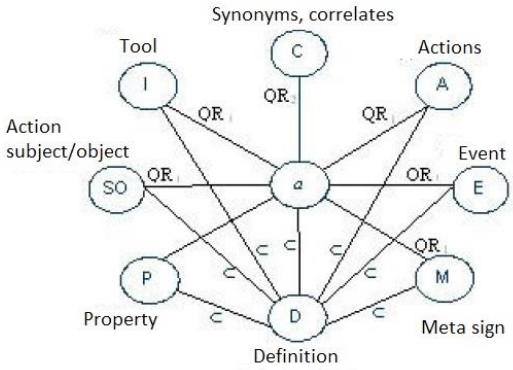


Fig. 2. Graphic representation of "action" sign

Here the elements of *t* and *a* - the term name, the type of object and the conceptual view of nature: material or immaterial. Most of the other data elements of the vectors represent the relations between the terms. Set of substantial definitions (D) and methods metalinguistic representation (M) in this case are not considered. Detailed design of categorical signs presented in [1].

Concerning types of conceptual relationships qualitative relations between ontology terms can be classified as follows:

- hierarchy (abstract-concrete area)
  - Concept-Concept (*T*)
  - Action-Action (*A*)
  - Property-Quantity (*Q*)
- aggregation (attachment area)
  - Concept-Concept (*T*)
  - Action-Action (*A*)
  - Concept-State (*S*)
  - State-Event (*E*)
  - <term>-Property (*P*)
- functional (processuality area)
  - Concept-Action (*A*, *I*)
  - Concept-Action \ Action-Concept (*SO*)
  - Action-Event (*E*)
  - Event-State (*S*)
  - Property-Quantity (*Q*)
- semiotic relations (area of content and form) relate to methods of metalinguistic representation (M)

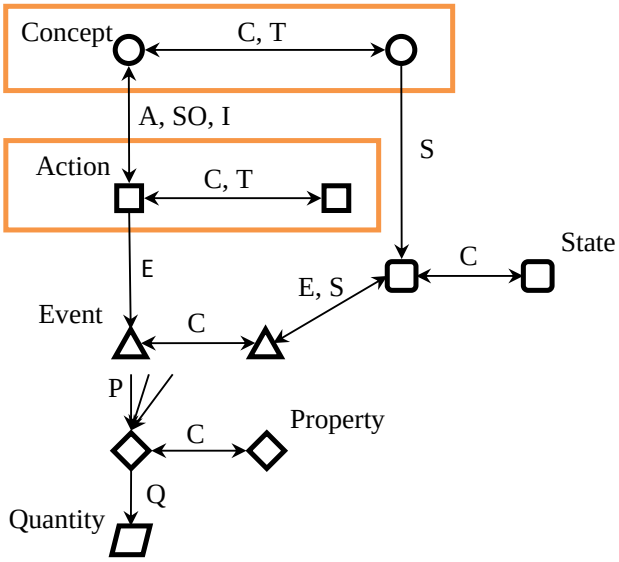


Fig. 3. Diagram of the relations between the terms of categorical signs

Also present quantitative relations in the ontology (the identity of the scope and correlation, C). Analysis of these relations will determine the consistency of concepts and relations of the ontology.

Domain ontology contains a structured open data, which makes it possible to assess the application of certain properties of formal concept analysis methods. Our study is to analyze the relations within these structures through the use of concept lattices.

3 The approach to ontology evaluation

3.1 The purpose of the analysis

The purpose of the analysis of this approach is **the completeness of the ontology relations**. This property shows the extent to which knowledge about the relations between domain terms displayed in the ontology.

To evaluate this property is necessary to determine whether the ontology relations complete and consistent. In this paper, we consider only the qualitative relations.

3.2 Description of the approach

The basis of the analysis is to find inconsistencies between the grid concepts, built on a certain relation, and ontology relations.

Analysis in accordance with the approach consists of several sequential steps (figure 4):

1. Select the type of term relation that you want to analyze. Every relation type has its semantic meaning, so the result of the analysis is interpreted according to the selected type.
2. Construction of concept lattices. On certain relations between the concepts of the lattice constructed ontology terms where terms are considered categories are taken as objects and attributes. The theme of constructing a formal context and concept lattice is mentioned in a large number of works devoted to the FCA, such as [10-13] and etc. This theme has been well studied. Depending on the relation type it is possible to use different methods of constructing a formal context to maximize the effectiveness of analysis.
3. Search lattice inconsistencies and the structure of the ontology relations. Here are compared with corresponding lattice of concepts relating to the structure of the ontology. A search of all relations, which are absent in the resulting lattice or structure of ontology relations. When constructing lattices on the basis of terms between the different categories of objects and attributes are taken in such a lattice terms of different categories.
4. Analysis of the inconsistencies. This analysis is performed by an expert, however, can be made automatically concluded terms with the greatest discrepancy coefficient, ie, terms which are associated with greater inconsistencies.

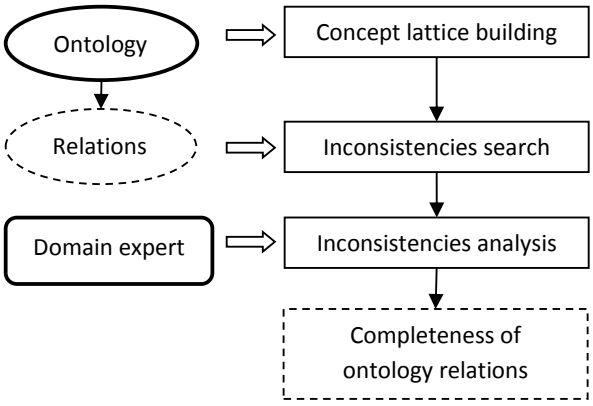


Fig. 4. Sequence of analysis steps

By the terms of categorical signs exist qualitative relations that define some hierarchy of terms relative to each other. However, only the terms of "Concept" and "Action" categories are qualitative relations with the terms of its category.

The relations between the terms of these categories can be divided into two types: relations of one category (Concept-Concept, Action-Action) and relations between categories (Concept- Action, Action-Concept). When looking for inconsistencies using both types of relations, and they have different effects on the result of the analysis.

3.3 Relations of one category

"Concept-Concept" and "Action-Action" relations have different semantics. However, they are similar to the structure, so the relations are equivalent to the analysis of terms.

Consider the example of such relations "Class-Kind" between the terms of the "Concept" category of ontology. Table 1 provides a formal context received on the relation.

In the construction of the formal context the known methods of its constructing can be used. In this example, we use a simple method: all terms that do not take the role of "Class" in any relations are formal objects, and other terms are formal attributes.

Table 1. Example of formal context

G\M	hoofed	herbivorous	overland	predator
Cow	X	X	X	
Rabbit		X	X	
Wolf			X	X
Piranha				X

Figure 3 shows a lattice of concepts on the formal context.

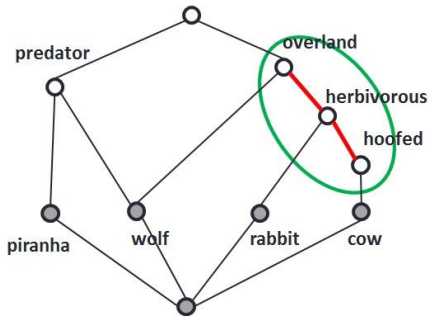


Fig. 5. Example of concept lattice

For example, in this example, fragment structure of " Class-Kind " relations of ontology is as follows:

- overland. Connected with:
  - cow
  - rabbit
  - wolf
- herbivorous. Connected with:
  - rabbit
  - cow
- hoofed. Connected with:
  - cow

From this it follows that the relation between the terms "overland" and "herbivorous" and the relation between the terms "herbivorous" and "hoofed" obtained during the construction of the lattice are absent in the source ontology (relations are marked in Figure 3). Thus, we can infer the probability that the set of relations of the ontology is incomplete. Found relations probably must be included in the ontology.

Let the set of relations of a particular type of source ontology is T_O, and the set of relations derived from a concept lattice of relations of the same type is T_R. Then the set of inconsistencies relations of one category is defined as

$$N_T : T_O \setminus T_R \cup T_R \setminus T_O. \tag{3}$$

However, it should be separated by a set of lattice inconsistencies ($N_{TR} : T_R \setminus T_O$) and ontology inconsistencies as they may have a different weight in determining the completeness of the ontology relations.

As a result, we get a lot of incredible inconsistencies. Such inconsistencies can be a great multitude, which may confuse the expert. Therefore, relations between categories should be considered.

3.4 Relations between categories

"Concept-Action" and "Action-Concept" relations have different semantics. However, they are similar to the structure, so the relations are equivalent to the analysis of terms.

Consider the example of such "Concept-Action" relation of ontology. Table 2 presents a formal context received on the relation. Unlike the previous example, where the objects and attributes are terms of the same category, in this case the objects are all terms of "Concept" category, and attributes are terms of "Action" category.

Table 2. Example of formal context

G\M	moos	jumps	eats meat	floats
Cow	X	X		
Rabbit		X		
Wolf		X	X	
Piranha			X	X
overland	X	X	X	
herbivorous	X	X		
Hoofed	X	X		
predator		X	X	X

Figure 4 shows concept lattice of the formal context.

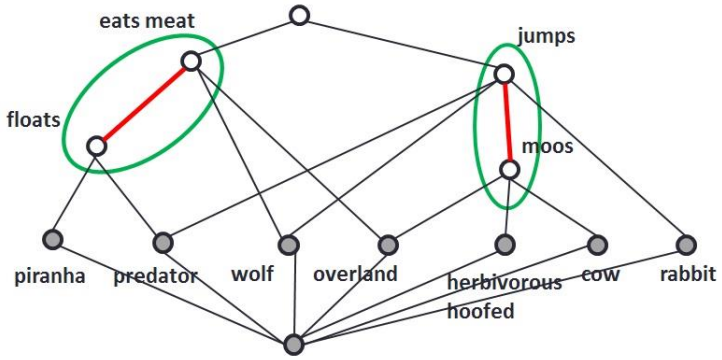


Fig. 6. Example of concept lattice

Assume in this example in the initial ontology actions "moos", "jumps", "eats meat" and "floats" are not connected qualitative relations between each other.

It follows from this relation between the terms "eats meat" and "floats" and the relation between the terms "jumps" and "moos" are absent in the source ontology and may be included therein (the relation of marked in Figure 4).

Let the union of qualitative relations of one category of original ontology is $\cup T_{Oi}$, and the set of relations derived from the concept lattice of relations of the same type is A_R.

Then the set of inconsistencies of the relation is defined as

$$N_A : (\cup T_{Oi}) \backslash A_R \cup A_R \backslash (\cup T_{Oi}). \quad (4)$$

However, it should be separated by a sets of lattice inconsistencies ($N_{AR} : (\cup T_{Oi}) \backslash A_R$) and ontology inconsistencies ($N_{TO} : A_R \backslash (\cup T_{Oi})$) as they may have a different weight in determining the completeness of the ontology relations.

Unlike lattices of relations of one category in this case is not specified the type of relations based on the qualitative, which searches for inconsistencies. On the basis of the terms of relations with the other categories of construction abstract terms this category hierarchy.

Because of lack of a particular type of relations such inconsistencies have little weight in the analysis, however, together with the inconsistencies obtained by relations of one category define a more detailed analysis of the completeness of the ontology relations.

Thus, inconsistencies, which are available to the expert for consideration, determined by

$$N : N_T \cap N_A. \quad (5)$$

As a result, the expert receives the set is not appropriate for the two parameters of relations. This allows it to draw a conclusion about the completeness of the ontology relations.

4 Conclusion

Formal Concept Analysis provides additional opportunities for analysis of the ontology of the model. Formal context and concept lattice allows sharing the concepts of the ontology of individual relations, which provides a more detailed analysis of ontological knowledge.

The presented approach allows us to evaluate the coherence of concepts and relations of ontology based concept lattice. Lattice allows you to identify the logical dependencies based on the relations of one category or hidden depending based relations with the terms of different categories.

To develop an accurate and effective method for the analysis of completeness of ontology relations requires further research relations and properties of the ontology. In this paper, we considered only the basic relations on the terms of the "Concept" and "Action" categories. For complete analysis there is needed for further study of the relations and the inclusion in the analysis of the terms of other categories.

At the moment, the approach is still under development. For further development of the approach required to examine all possible relations between the terms of ontology. It is necessary to determine the exact relations between the types of relations for a full analysis of inconsistencies in the ontology and the completeness of the ontology relations.

References

1. Klesh'eva A.S., Shalfeeva E.A.: Classification of ontology properties. Ontologies and their classification. In: IACP, 2005, 20 p., Preprint, Vladivostok (2005)
2. Naihanova L.V.: The technology of building methods for automatic construction of ontologies using genetic and automata programming. Monograph. BSC SB RAS, 244 p., Ulan-Ude (2008)
3. Ganter B., Wille R. Elliot H. Applied Lattice Theory: Formal Concept Analysis. – http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.9907
4. Rudolf Wille: Formal Concept Analysis as Applied Lattice Theory. In: CLA 2006, pp.42-67 (2006)
5. Dambaeva S.V., Merdygeyev B.D.: Ontology structures: New Information Technologies and Systems (NITS-2014). In: 11th International scientific–technical conference, pp. 222-227, PSU, Penza (2014)
6. Shalfeeva E.A., Gribova V.V. The internal properties of ontology. In: SICPRO'05. IV International conference. pp. 1109-1128, Moscow (2005)
7. Uta Priss: Knowledge Discovery in Databases Using Formal Concept Analysis. In: Bulletin of the American Society of Information Science 27, vol. 1, p. 18-20 (2000)
8. Dahlberg Ingetraud: Knowledge Organization: its scope and possibilities / Knowledge Organization: problems and trends // Proceedings of Conference reports, Moscow (1993)
9. Nikitina S.E.: Semantic analysis of the language of science. In linguistics material. Monograph. Book House "LIBROKOM", 146 p., Moscow (2010)
10. Stumme G., Maedche A.: FCA-merge: Bottom-Up Merging of Ontologies // IJCAI'01 Proceedings of the 17th international joint conference on Artificial intelligence, 2001 - https://www.researchgate.net/publication/2475502_FCA-Merge_Bottom-up_merging_of_ontologies
11. Stumme G., Cimiano P., Hotho A., Tane J.: Conceptual Knowledge Processing with Formal Concept Analysis and Ontologies. – http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.94.1946
12. Wille R.: Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts In: I. Rival (ed.): Ordered sets. Reidel, Dordrecht-Boston, p. 445-470 (1982)
13. Uta Priss: Lattice-based Information Retrieval Knowledge Organization. – http://www.upriss.org.uk/papers/ko00.pdf

Author Index