Fourth Conference on Software Engineering and Information Management

CEUR Workshop Proceedings
Fourth Conference on Software Engineering and Information Management (SEIM-2019) (full papers)
Saint Petersburg, April 13, 2019
Yurii Litvinov, Peter Trifonov (editors)
This volume contains ten selected papers originally presented at the Fourth Conference on Software Engineering and Information Management (SEIM-2019), which was held in Saint Petersburg, Russia, on April 13, 2019. These papers were selected in thorough single-blind reviewing process.


Copyright © 2019 for the individual papers by the papers’ authors. Copying permitted for private and academic purposes. This volume is published and copyrighted by its editors.
# Table of Contents

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message from the Editors</td>
<td>4</td>
</tr>
<tr>
<td>SEIM-2019 Organization</td>
<td>5</td>
</tr>
<tr>
<td>A hybrid convolutional and recurrent network approach for conversational AI in spoken language understanding</td>
<td>6</td>
</tr>
<tr>
<td>Bassel Zaity, Hazem Wannous, Igor Chernoruckiy, Pavel Drobintsev, Zein Shaheen and Vadim Pak</td>
<td>6</td>
</tr>
<tr>
<td>The DSL for composing functions for FaaS platform</td>
<td>13</td>
</tr>
<tr>
<td>Nikita Gerasimov</td>
<td></td>
</tr>
<tr>
<td>Path planning for UAV search using growing area algorithm and clustering</td>
<td>20</td>
</tr>
<tr>
<td>Alexander Fedorov</td>
<td></td>
</tr>
<tr>
<td>Machine Learning Methods for Earthquake Prediction: a Survey</td>
<td>25</td>
</tr>
<tr>
<td>Alyona Galkina and Natalia Grafeeva</td>
<td></td>
</tr>
<tr>
<td>Construction of polarization kernels of size 16 for low complexity processing</td>
<td>33</td>
</tr>
<tr>
<td>Grigori Trofimnik and Peter Trifonov</td>
<td></td>
</tr>
<tr>
<td>A Quantitative Study of Two Matrix Clustering Algorithms</td>
<td>40</td>
</tr>
<tr>
<td>Alexander Slesarev, Viacheslav Galaktionov, Nikita Bobrov and George Chernishev</td>
<td>40</td>
</tr>
<tr>
<td>Designing a DBMS Development Course with Automatic Assignment Evaluation</td>
<td>48</td>
</tr>
<tr>
<td>Viacheslav Galaktionov and George Chernishev</td>
<td></td>
</tr>
<tr>
<td>Word embedding in form of symmetric and skew-symmetric operator</td>
<td>54</td>
</tr>
<tr>
<td>Ekaterina Koshchenko and Igor Kuralenok</td>
<td></td>
</tr>
<tr>
<td>Medical Images Research Framework</td>
<td>60</td>
</tr>
<tr>
<td>Sabrina Musatian, Alexander Lomakin and Angelina Chizhova</td>
<td></td>
</tr>
<tr>
<td>Implementation of “smart greenhouse” visual programming tool using deep metamodeling</td>
<td>67</td>
</tr>
<tr>
<td>Elizaveta Kuzmina and Yuri Litvinov</td>
<td></td>
</tr>
</tbody>
</table>
Message from the Editors

The Fourth Conference on Software Engineering and Information Management (SEIM-2019) opens its doors to young researchers and practitioners in different areas of computer science and software engineering, providing an opportunity to present their research, discuss state-of-the-art technology and engage in useful networking. As before, we consider SEIM to mainly focus on researchers who are just starting out their scientific careers, and hope to ease their introduction to the conference process. On the other hand, SEIM might also be of interest to more experienced researchers, who are aimed at sharing their research with a wider scientific community.

The conference welcomes submissions on a wide range of topics, including but not limited to:

- Algorithms and data structures
- Big data
- Cloud systems
- Coding theory
- Compilers
- Crowdsourcing
- Data storage and processing
- Development management
- Digital signal processing
- Distributed systems
- E-commerce / e-government
- Empirical software engineering
- High-performance computing
- Information retrieval
- Information security
- Intelligent data analysis
- Internet of Things
- Machine learning
- Mobile systems
- Modelling
- Natural language processing
- Networks and telecommunications
- (Non-)relational databases
- Operating systems
- Programming languages
- Recommendation systems
- Robotics
- Semantic web
- Social networks
- Software analysis
- Software testing
- Software verification
- Software virtualization
- Software-defined networks
- Theoretical computer science
- Visual languages

This year we received 64 papers, each reviewed by at least 3 members of the Program Committee, of which 10 were selected for publication in CEUR-WS.org, 10 for indexing in RSCI, and 7 were accepted as talk-only to allow the authors to receive valuable feedback about their research. We would like to thank the members of our Program Committee for their continuous work and contribution to the success of our conference.

These proceedings include the SEIM-2019 papers, which were selected by the Program Committee for publication in CEUR-WS.org. These papers passed not only the original review procedure, but also an additional round of post-review with the conference feedback. We thank the authors for their submissions to SEIM-2019 and hope to see them in the future.

Furthermore, we would also like to thank Tatiana Mironova, Sergey Zherevchuk and Svyatoslav Mikhailov for their great help in organizing the conference, Computer Science Center for hosting the event, and JetBrains Research for their overall support. The additional information about the SEIM conference series can be found on the conference website at: [http://2019.seim-conf.org/](http://2019.seim-conf.org/)

Yurii Litvinov, Peter Trifonov (editors)
The conference was organized jointly with Computer Science Center and supported by JetBrains Research.

Steering Committee

Dmitry Bulychev / St. Petersburg State University
Vladimir Itsykson / St. Petersburg Polytechnic University
Andrey Ivanov / JetBrains
Iakov Kirilenko / St. Petersburg State University
Kirill Krinkin / St. Petersburg Electrotechnical University
Boris Novikov / St. Petersburg State University

Program Committee Chairs

Yurii Litvinov / St. Petersburg State University
Peter Trifonov / St. Petersburg Polytechnic University

Program Committee

Marat Akhin / Saint Petersburg Polytechnic University
Mikhail Belayev / Saint Petersburg State Polytechnical University
Danii Berezun / JetBrains and Saint Petersburg State University
Sergey Bezzateev / Saint Petersburg University of Airspace Instrumentation
Natalia Bogach / State Polytechnical University
Dmitry Bulychev / Saint-Petersburg State University
Timofey Bryksin / Saint Petersburg State University
Dmitry Bugaychenko / Saint-Petersburg State University
George Chernishev / SPbU
Mikhail Glukhikh / JetBrains
Natalia Grafeeva / Saint-Petersburg State University
Semyon Grigorev / Saint Petersburg State University
Semyon Grigorev / St. Petersburg State University
Igor Malishev / Saint-Petersburg State Polytechnical University
Vladimir Itsykson / SPbSU
Aleksandr Kapitonov / TTMO University
Iakov Kirilenko / Saint Petersburg State University
Vladimir Kovalenko / Delft University of Technology
Kirill Krinkin / Open Source and Linux Lab
Svetlana Lazareva / Raidix
Dmitry Luciv / Saint-Petersburg State University
Igor Malishev / Saint-Petersburg State Polytechnical University
Dmitry Mordvinov / Saint-Petersburg State University
Boris Novikov / National Research University Higher School of Economics
Andrei Ovchinnikov / SUAI
Oleg Pliss / Oracle
Anton Podkopaev / Saint Petersburg State University
Stanislav Sartasov / SPbSU
Yuri Senichenkov / Saint Petersburg State Polytechnical University
Kirill Smirnov / SPbSU
Darja Solodovnikova / University of Latvia
Sergey Stupnikov / Institute of Informatics Problems, Russian Academy of Sciences
Dimitri Timofeev / Peter the Great St.Petersburg Polytechnic University
Dmitry Tsitelov / Devexperts, LLC
Andrey Turlakov / Saint Petersburg University of Airspace Instrumentation
Ekaterina Verbiktskaya / Saint Petersburg State University
Mikhail Zymbler / South Ural State University
A hybrid convolutional and recurrent network approach for conversational AI in spoken language understanding

1st Bassel Zaity
Graduate School of Software Engineering
Peter the Great St.Petersburg Polytechnic University (SPbPU)
Saint Petersburg, Russia
bassel.zaity@gmail.com

2nd Hazem Wannous
IMT Lille Douai
CRISAL UMR 9189, University of Lille
Lille, France
hazem.wannous@univ-lille.fr

3rd Zein Shaheen
Computer Intelligent Technologies
Peter the Great St.Petersburg Polytechnic University (SPbPU)
Saint Petersburg, Russia
shahin.z@edu.spbstu.ru

4th Igor Chernoruckiy
Graduate School of Software Engineering
Peter the Great St.Petersburg Polytechnic University (SPbPU)
Saint Petersburg, Russia
igcher@spbstu.ru

5th Pavel Drobintsev
Graduate School of Software Engineering
Peter the Great St.Petersburg Polytechnic University (SPbPU)
Saint Petersburg, Russia
drob@ics2.ecd.spbstu.ru

6th Vadim Pak
Computer Intelligent Technologies
Peter the Great St.Petersburg Polytechnic University (SPbPU)
Saint Petersburg, Russia
vadim.pak@cit.icc.spbstu.ru

Abstract—The deep learning revolution has an impact on almost all parts of our life, it brought us improved momental machine translators, modern human-like conversation voice assistant like Siri, Alexa, Alisa. This revolution had become truth because of deep learning methods which improved multiple processing layers to learn a hierarchical representation of data, and have achieved the state-of-the-art results in many lives domains. In this paper, we are focusing on one of the most famous NLP (Natural language processing) problems which is slot filling to approach the state-of-the-art results on the ticketing problem to make the Spoken Dialogue systems work more efficiently. We propose a hybrid architecture, as a combination of a Recurrent Neural Network and a Convolutional Neural Network models, for Slot Filling in Spoken Language Understanding. In particular, our network model is built from stacked units of 1-dimensional CNN (Convolutional Neural Network) across the temporal domain, which are used to train an RNN (Recurrent neural network) layer to model dependencies in the temporal domain. Experimental tests show extensive comparisons between different models for NER (Named Entities Recognition). Results demonstrate the effectiveness of hybrid models that combine benefits from both RNN and CNN architecture compared over distinct RNN and CNN models and also compared with other traditional models. Experimental results show that our model achieves F1-score of 95.11 on benchmark ATIS dataset.

Index Terms—SLU, slot-filling, Hybrid CNN and RNN, Deep learning

I. INTRODUCTION

The methodological revolution in spoken language research had been started about 20 years ago when the machine learning algorithms started to take place in the programmer society. However, the last five years brought the real change after the new deep learning architectures, which led to a new level of solutions and the Spoken Dialogue Systems (SDS) is one of the fields which had really improved recently. SDS and chatbots are taking a wider place day by day in the scientific conferences as a case study. They already have great commercial potential according to the changing of the way humans interact with machines. The improvement of deep learning in general, and the Natural Language Processing (NLP) researchers in special, led to place a lot of difficult problems under the microscope, and the research teams over the world trying to test different architecture models to get the state-of-the-art results to solve these problems. In our days, the importance of chatbots has increased, most websites tend to have their own chatbots to communicate with customers and facilitate their work. The goal of such bots is to know users needs and give responses in their natural language. This will lead to a better understanding of the users queries when communicating with the users in a natural way throw these chatbots. It will also help to ask the users about whatever missing points they have to bring the best accurate answers, such assistants could help disabled people and bring more solutions to the market to build a more intelligent world.

The implementation of a voice assistant comes with different parts, as speech to text and text to speech models, but the most challenging part comes in the task of NLP to extract
the needs of the user and to know his intent from the conversation. The processing pipeline comes here into two parts, intent classification and slot filling after the intent is known. At this stage, the bot needs to generate a response to the user and give feedback about whatever missing data there are. The whole system that organizes this process is the dialogue manager which processes the users input, extract the meaning and generates the desired response. From a research perspective, the design of spoken dialogue systems provides a number of significant challenges, as these systems depend on solving several difficult NLP and decision making tasks, and combining these into a functional dialogue system pipeline [1].

Intent detection and slot filling are usually processed separately. Intent detection can be treated as a semantic utterance classification problem, and popular classifiers like support vector machines (SVMs) [2] and deep neural network methods [3] can be applied. Slot filling can be treated as a sequence labeling task. Popular approaches to solving sequence labeling problems include maximum entropy Markov models (MEMMs) [4], conditional random fields (CRFs) [5], and recurrent neural networks (RNNs) [6] [7] [8]. Joint model for intent detection and slot filling has also been proposed in literature [9] [10]. Such joint model simplifies the spoken language understanding (SLU) system, as only one model needs to be trained and fine-tuned for the two tasks.

This work focuses on the slot-filling part by building a model that extracts information from text in a reliable way. Before the era of deep learning the task of Named Entity Recognition (NER) was solved using grammars-based models and rule-based approaches, these models have proven to achieve good results in terms of precision but fail to capture all human-text varieties and thus the recall will be bad. Probabilistic approaches came with models built on HMM, which were state-of-art for many years and achieved an impressive achievement. With the recent revolution, many deep learning methods has replaced traditional previous ones and pushed state-of-art for these tasks. Recurrent Neural Networks (RNN) models have replaced models based on HMM, that is NER achieved the same task in a simpler way and deep RNNs are able to capture complex representations for the input. The problem with such models was that they need to handle the input token-by-token in sequence. Therefore, such structures could not be parallelized and the models will be slow to train and inference if the neural network structure is deep. Convolution Neural Networks (CNN) added a way to extract relations between tokens by mixing them in a way similar to extracting n-grams in the traditional NLP tasks. Such architectures that contain CNN could be optimized by parallelization so adding a convolutional layer could reduce the complexity and control the size of the neural network. In this paper, we discuss different approaches to solve slot filling for ticketing task as a NER problem, and showed different architectures that contain distinct RNN, CNN or hybrid architectures ones. We conducted many experiments with different values of the hyper-parameters and different optimization methods.

II. RELATED WORK

Rule-based approaches are done manually, at first you all needed roles should be written need to achieve the goal, this operation is time-consuming and therefore not so efficient, it will be notable that the recall is not very nice because its so difficult to write all the varieties, but the positives of ruled-based approaches the precision will be quite high [11]. The most widely used formal system for modeling constituent structure in English and other natural languages is the Context-Free Grammar or CFG. A context-free grammar consists of a set of rules or productions, each of which expresses the ways that symbols of the language can be grouped and ordered together, and a lexicon of words and symbols [12].

In machine learning methods, we need a dataset of text with markup, in this dataset, each word should be assigned to a tag, this problem is known as slots filling problem. The first which we should do is making some Feature engineering, for example, see whether the word is capitalized or it is a name of a city, some cities consists of two words, maybe you check the previous or the next words (context). Probabilistic modeling and Conditional Random Field not only assume that features are dependent on each other but also considers the future observations while learning a pattern [21]. This combines the best of both HMM and MEMM. In terms of performance, it is considered previously to be the best method for entity recognition problem. Another paper studied the comprehensive investigations of RNNs for the task of slot filling in SLU. They implemented and compared several RNN architectures, including the Elman-type and Jordan-type networks with their variants [18]

III. DEEP LEARNING METHODS

A. Recurrent Neural Network RNN

Recurrent Neural Networks “Fig. 1” are used for sequence modeling, it accepts input $x_t$ at time step $t$ and a hidden state $h_t$ and use this hidden state to produce output $y_t$, and this hidden state will be passed to the next time step. So, we can think of the hidden state as a summary of the previous inputs to the neural network, we use activation function such as $\text{tanh}$ or ReLU to calculate hidden state. Output $y_t$ is the prediction of the next tag, it would be a vector of probabilities across our vocabulary, the following formulas explain the general form of RNN:

$$h_t = f(Ux_t + Wh_{t-1})$$  \hspace{1cm} (1)

$$y_t = \text{softmax}(Vh_t)$$  \hspace{1cm} (2)

Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU) are used as RNN units, these units can capture long term dependency. The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates. Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation [19].
The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means let nothing through, while a value of one means let everything through! An LSTM has three of these gates “Fig. 2”, to protect and control the cell state. The following formulas explain how does LSTM cell work:

\[
f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \\
i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \\
C_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \\
f_t . C_{t-1} + i_t . C_t \\
o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \\
o_t . \tanh(C_t)
\]

In our experiments, we used both GRU and LSTM units and compared between them. Other sequence architectures like Encoder-decoder architecture could be used to solve this task, at first the whole input will be encoded into hidden representation (encoder), and then this hidden representation is used to produce sequence of tags (decode). Some architectures use attention mechanism to give attention to parts of the input sequence and use these information to produce the output token.

**B. Convolution Neural Network for sequences**

RNNs operate sequentially, the output for the second input depends on the first one and so we can’t parallelize an RNN. Convolutions have no such problem, each patch a convolutional kernel operates on is independent of the other, meaning that we can go over the entire input layer concurrently. Convolutions grow a larger receptive field as we stack more and more layers. That means that by default, each step in the convolutions representation views all of the input in its receptive field, from before and after it “Fig. 4”. In our experiments we used 1D convolution to mix the tokens and extract relations between the consequence tokens, it is equivalent to n-gram relation where n is the size of the used filter, for example: if we care about the last 3 tokens we use filter size 3. Using CNN will result in some benefits, it runs faster than RNN and beats RNN in some tasks. If we divide convolution output into two parts, A and B, one of which will gate the other through element-wise multiplication, where A is linear and B through sigmoid, we get GLU (gated linear unit). Here we increased receptive field as it is shown in the following formula:

\[
A = (X.W + b) \\
B = \sigma(X.V + c) \\
h_t(x) = A \oplus B \\
h_t(x) = (X.W + b) \oplus \sigma(X.V + c)
\]
C. Hybrid model CNN RNN

This model combines the benefits of both CNN and RNN, where RNN helps to capture the dependencies between tokens in the users query, using LSTM or GRU units will have resulted in a model that captures long-range dependencies between tokens using memory cell in their architecture. CNN will help with mixing the consequence tokens and extract relations between them [14]. In the task of slot filling, the hybrid architecture contains several convolution layers stacked with the same padding and the output of these layers will be the input for RNN layers as in Fig. 5, we can also stack several RNN layers. After these RNN layers, there will be a dense layer with softmax activations, this layer represents the output of the network.

IV. EXPERIMENTS

A. Dataset

ATIS (Airline Travel Information System) corpus (Tur et al., 2010) is one of the main data resources used in many studies over the past two decades for SLU research in spoken dialog systems e.g. [15] [16] [17]. Two primary tasks in SLU are intent determination (ID) and slot filling (SF). The dataset contains audio recordings of people making flight reservations. The training set contains 4,478 utterances and the test set contains 893 utterances. We use another 500 utterances for development set. There are 120 slot labels and 21 intent types in the training set [22].

The IOB format (inside, outside, beginning) is a common tagging format for tagging tokens in a chunking task in computational linguistics. The B- prefix before a tag indicates that the tag is the beginning of a chunk, and an I- prefix before a tag indicates that the tag is inside a chunk. The B- tag is used only when a tag is followed by a tag of the same type without O tokens between them. An O tag indicates that a token belongs to no chunk.

The Table I shows an example in the ATIS dataset, with the annotation of slot/concept, named entity, intent as well as domain. The latter two annotations are for the other two tasks in SLU: domain detection and intent determination. We can see that the slot filling is quite similar to the NER task, following the IOB tagging representation, except for a more specific granularity.

1) Training Details: In training, we compared between different models for NER (Named Entities Recognition) system, all the models were trained using 100 epochs. We tuned our models using different dropout values (0.1, 0.25, 0.5) and we used different optimization methods (ADAM, RMSPro, SGD). For the embedding layer, we represent each token by a vector of size 100, and for our choice for the convolution layer we used 64 filters of size 5 and used ReLU as an activation function. The hidden size of the GRU/LSTM unit is 100 “Fig. 5”.

Our architecture will go as following, input layer which is a sequence of tokens represented by indices using bag of words, embedding layer will represent each token with a vector, the vector size is a hyperparameter for the network, this embedding layer is followed by one of the main choices of the layers discussed above, recurrent neural network, convolutional neural network or a hybrid model which contains layer of CNN followed by layer of RNN.

2) Evaluation Metrics: For evaluation, we computed precision, recall and F1 score for training and validation sets, and we picked the model with the best value of the F1 score. For Slot filling, the error rate can be computed in two ways: The more common metric is the F-measure using the slots as units. This metric is similar to what is being used for other sequence classification tasks in the natural language processing community, such as parsing and named entity extraction. In this technique, usually the IOB schema is adopted, where each of the words is tagged with their position in the slot: beginning (B), in (I) or other (O). Then, recall and precision values are computed for each of the slots. A slot is considered to be correct if its range and type are correct. The F-Measure is defined as the harmonic mean of recall and precision:

$$F1-Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \quad (17)$$

where:

$$Recall = \frac{\#correct \ slots \ Found}{\#true \ slots} \quad (18)$$

$$Precision = \frac{\#correct \ slots \ Found}{\#found \ slots} \quad (19)$$

B. Results

During evaluation process we focused on the difference between the use of different architectures of neural networks, we compared also between different optimization methods for the best neural network structure and at the end we included
a comparison based on the type of recurrent unit used in the model. We concluded the experiments 25 times, and we took the mean of the samples and calculated the standard error. We reported our results in the tables.

Our results show that hybrid architectures perform better than other pure RNN or pure CNN models Table II, when we used dropout 0.25 and RMSProp optimization method, we got F1-score 95.04 for hybrid model compared with 91.16 for convolution model and 93.07 for recurrent model.

Our results show also that the use of RMSProp resulted in the best models according to F1-score metrics Table III, under the same dropout 0.25 and hybrid model, we got F1-score equals to 95.04 for RMSProp compared with 94.83 when we used ADAM optimization model, and 94.44 when we used SGD.

Result show that the effect the Hybrid structure Convolution1D and RNN/GRU with dropout 0.25; RMSProp gave the best F1-score 95.11 comparing with different levels of dropout on the same architecture Table IV. Based on the recurrent unit used in our experiments, GRU based hybrid methods with F1-score 95.04 compared with LSTM based hybrid models with F1-score 94.67, GRU units improved the score by 0.37% Table V. Our results show that the hybrid CNN/RNN-based models outperform Bi-dir. Jordan-RNN baseline by 1.13% on the ATIS benchmark Table VI.
TABLE IV
COMPARISON BETWEEN HYBRID STRUCTURES BASED ON THE USED RECURRENT UNIT

<table>
<thead>
<tr>
<th>Structure description</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid structure Conv1D and RNN/GRU with dropout 0.25; RMSProp</td>
<td>94.47</td>
<td>95.61</td>
<td>95.04</td>
<td>94.89 ±0.15</td>
</tr>
<tr>
<td>Hybrid structure Conv1D and RNN/GRU with dropout 0.1; RMSProp</td>
<td>94.29</td>
<td>95.31</td>
<td>94.82</td>
<td>94.42 ±0.28</td>
</tr>
<tr>
<td>Hybrid structure Conv1D and RNN/GRU with dropout 0.25; RMSProp</td>
<td>94.47</td>
<td>95.61</td>
<td>95.04</td>
<td>94.89 ±0.15</td>
</tr>
<tr>
<td>Hybrid structure Conv1D and RNN/GRU with dropout 0.5; RMSProp</td>
<td>93.3</td>
<td>94.6</td>
<td>93.95</td>
<td>93.25 ±0.43</td>
</tr>
</tbody>
</table>

TABLE V
COMPARISON BETWEEN DIFFERENT DEEP LEARNING STRUCTURES BASED ON DROPOUT VALUE

<table>
<thead>
<tr>
<th>Structure description</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid structure Conv1D and RNN/GRU without dropout; RMSProp</td>
<td>94.98</td>
<td>95.47</td>
<td>95.11</td>
<td>94.69 ±0.47</td>
</tr>
</tbody>
</table>

V. CONCLUSION

This paper addresses the problem of slot filling in Spoken Language Understanding. In particular, we focused on slot tagging without paying attention to the other intent classification part. We formulated our learning architecture as a hierarchy of spatial CNN features followed by the RNNs to model dependencies in the temporal domain. Experimental results on the ATIS dataset consistently demonstrated the effectiveness of the proposed approach. It is good to mention that combined models that solve the two tasks at the same time could be implemented and these models had proven to lead to better performance. But still, in the way to implement a full chatbot, we will need to generate human-like text in response to users input. In future work, we intend to explore the incorporation of attentional mechanism in our model, which could provide additional information to the slot label prediction, and learn our architecture using another data-sets to generalize the results.

REFERENCES


The DSL for composing functions for FaaS platform

Nikita Gerasimov
Mathematics and Mechanics Faculty
Saint Petersburg University
St. Petersburg, Russia
n.gerasimov@2015.spbu.ru

Abstract—This article describes the problems that occur when using the Function as a Service model: the complexity of the centralized description of the separate function interaction within the whole system and the possibility of dependent components interface divergence during a process of their development. We propose a domain-specific language called Anzer as a solution to these problems which enables to describe the types of data transmitted, the composition of functions, the semantics of their interaction and logic of error handling. To check the consistency and compliance of the declared and implemented types, software has been produced making it possible to automate the creation of new functions and maintain their integration with others at all stages of system development and support. The language and software in combination with each other prevent errors associated with a mismatch between input data format expected by the function and actually transmitted one. All this enables to simplify and speed up the process of developing systems based on the concept of Function as a Service.

Index Terms—serverless, faas, static typing, function composition, domain-specific language

I. INTRODUCTION

The development of the Internet, the emergence of a large amount of data and process control automation have led to an increase in the software modularity. An earlier approach to partitioning programs into components was service-oriented architecture (SOA), which is characterized by getting particular sets of functions separated into independent modules. Each module is responsible for a certain range of tasks. The interaction is carried out either using enterprise service bus or RPC API.

The next iteration of the development of modular distributed software architectures is microservice architecture. The approaches are very similar at first glance, but the difference is in details. Microservices usually tend to perform much fewer functions and have fewer dependencies, which simplifies scaling [1]. In addition, they provide one or more API methods rather than complex RPC interfaces covering most of the subject area.

Microservice architecture imposes certain requirements on the infrastructure: deploy automation, automation of testing, infrastructure for service discovery.

Without preliminary preparation of such an environment, developing large systems in such a paradigm leads to greater costs than developing a monolithic system [2].

In 2014 Amazon introduced AWS Lambda serverless platform (serverless computing platform) within Amazon Web Services cloud platform. The main logical unit in it is a function, which, in fact, is also a microservice, but with some differences [3]:

- A microservice is a standalone application having necessary libraries. A function is code that implements only necessary logic. Function start, its initialization, connection to a database are carried out by a platform, which means a function itself is not self-sufficient.
- A microservice can be run separately, and a function can often be started only within a framework of a serverless platform.
- A microservice typically runs in daemon mode, responding to incoming requests. A function is often run only when it is requested to execute it.

In summary, a function is similar to a microservice but often is not self-sufficient, does not store the state, uses capabilities of the platform to run and communicate. Such peer-to-peer computing platform called FaaS (Function as a Service).

In addition to Amazon, cloud FaaS platform services are provided by Google (“Google CloudFunctions”), Microsoft (“Azure Functions”) and IBM (“IBM Cloud Functions” based on the “Apache OpenWhisk” platform). In addition to cloud providers, there are self-hosted solutions: Apache OpenWhisk, Fission, OpenFaaS and other.

Simple applications with a web interface, or IoT (Internet of things) systems, often use separate, independent functions to perform actions when certain events occur. However, systems in the FaaS platform can be built through the composition of functions. For example, in a subsystem that processes data, one component passes intermediate results to another in a chain. One of the obvious applications of such a composition is ETL (Extract, Transform, Load) processes. Using a FaaS platform it is easy to implement a similar data conversion scheme if each step is allocated to one or more functions. The functions work in parallel, independently, do not store the state, return result of the work or an error message.

The benefits of serverless computing:

1) No need for infrastructure support (in case of using cloud-based FaaS solution providers).
2) No need to implement supporting code, such as logging, connecting to databases etc.
3) Simple scaling of individual components rather than a system as a whole.
4) No downtime costs because functions run on demand.
5) The need for a competent division of logic into modules, based on the concept definition.

The use of the solution mentioned above also has its drawbacks:

1) Lower transparency of the system compared to a monolithic application.
2) Difficulties with debugging of functions and a system as a whole.
3) There is no commercial solutions or standards in the field of systems testing that built in the FaaS platforms.
4) An unresolved issue of resources caching that should be initialized each time the function runs (for example, database connections).

Another problem that may arise during the development and support of such a system is the control of interaction and compatibility of individual functions within the whole system.

To solve this problem some FaaS cloud service providers make it possible to describe the order of functions run or their compositions. However, existing platforms do not provide proper type checking. There is a situation when a monolithic application is divided into separate logical blocks smaller even than microservices, and there is no way to guarantee these blocks will work together consistently and correctly. Condition may occur in which some of the components of the system provide the updated interfaces, and dependent components wait for outdated interfaces; that situation will result in an error in the process of operation. This fact is a disadvantage of serverless solutions in comparison with traditional methods of systems development: monolithic and service-oriented approaches.

In the article, we consider existing solutions for function composition in the FaaS platforms (sec. II), suggest a way to describe the composition of functions and automatically check their compatibility by means of DSL (sec. III) and a software complex that extends opportunities of the FaaS platform “Apache OpenWhisk” [4]. The use of such an extension is assumed to reduce number of errors associated with mismatch between function types during developing.

II. ALTERNATIVE SOLUTIONS

Serverless computing is a young approach, not yet widely known and not widely used. As a result, there are only a few solutions providing the composition of functions.

Amazon provides AWS State Machine with its own language, Amazon States Language describing a sequence of functions to run (AWS Step Functions) within a specific task [5]. Language and platform capabilities enable:

1) Define the order which functions should be started in.
2) Handle errors.
3) Set the number of data processing retries in case of error.
4) Run multiple processes in parallel.
5) Set the conditions for the launch of certain functions based on transmitted data.

Sequences of invoking functions are not functions themselves, they are state descriptions made by an external environment.

In addition to Amazon, IBM provides the ability to describe the composition of functions using a developed JavaScript library and “IBM Composer” functionality built in IBM Cloud Functions [5]. The solution also makes it possible to describe conditions, number of function retries and some other features. An important difference between IBM Composer and AWS State Machine is that composition in the former is also a function that can participate in composition.

Microsoft provides 2 mechanisms for the composition of functions in its “Azure Functions” platform: “Azure App Logic” and “Azure Durable Functions”. Just as “IBM Composer”, the mechanisms are built in the platform and provide an opportunity to describe conditions, cycles, number of retries, etc. [5].

If choosing among serverless self-hosted platforms, Fission and Fn enable to describe processes as the composition of functions using “Fission Workflows” and “Fn Flow” respectively [6]. These mechanisms also make it possible to describe the composition using conditional statements.

Project StdLib with FaaSlang provides a completely different way of using serverless technology. StdLib is platform-agnostic API gateway and serverless framework for FaaS enabling user to easily change serverless provider. FaaSlang provides an approach to specify a function’s input and output types. However, StdLib with FaaSlang does not ensure that the realized function fulfils its interface declarations. Moreover, the current state of the project does not support any language except JavaScript.

None of the solutions found enables to make sure in the minimal form that the function will start in the scheme of operation with parameters that are passed to it, i.e. to check in advance the compatibility of the data types of the interacting components. Therefore, we need a new alternative solution to describe the composition of functions which should provide the following features:

1) To describe the composition of functions, namely:
   • To describe the composition of one function with another, when the result of the first is passed to the second.
   • To describe error handling mechanism.

2) Make it possible to define the types of arguments and results for the functions involved in the composition and check their compatibility.

To meet paragraph 2 from the list of requirements, it is also necessary to have an extension for the FaaS platform which will manage functions building and deploying to ensure the type safety of the entire project because:

• If a function implements an interface different from the one the function declares, its compilation will be impossible.
• If the composition of a new version of the function is impossible with operating ones, its deployment will not happen.

We propose a domain-specific language called “Anzer”\(^1\)

\(^1\)https://github.com/tariel-x/anzer
as well as software that includes the language analyzer, the system of function building, interaction with the FaaS platform and the user interface as a solution to the problem mentioned above.

In addition, it is assumed that the language should not be highly specialized for use with a single FaaS platform. This means that the software package to be developed should be sufficiently versatile and modular to be able to adapt it to the new framework with minimal modifications.

III. THE LANGUAGE OF FUNCTIONS COMPOSITION

Functions in the FaaS platforms are triggered in case of need, perform the programmed action and pass the result forward, not keeping the state. This feature partly creates an affinity between the functions mentioned above and the concept of functions from some functional programming languages.

In addition, if we consider a set of functions as a single monolithic program, then, in the absence of a global runtime environment and variable changes, drawing an analogy with imperative programming languages is impossible. Functional approach, on the contrary, is characterized by the composition of independent functions, which means the result of the calculation of the previous function is applied to the next one. Also, programs written in a pure functional style do not contain mutable variables, and functions can be easily moved from one program to another.

Listed properties create an affinity between a functional programming style and systems built using the FaaS platforms. In this regard, the concepts of such functional programming languages as Haskell and PureScript were taken as a basis for the proposed language.

A. Type system

The implemented language supports both basic types (string, boolean etc.) and custom user-defined record types, as shown in listing 1.

```haskell
Listing 1. User-defined types
type Address = {
    house :: Integer
    street :: Maybe String
    city :: MinLength 10 String
    country :: String
}  
type Addresses = List Address
```

Also, the language supports extension of basic and user-defined types with the help of type constructors. List is the type constructor, that is, the function that converts Address type into List Address which is an array of addresses. MinLength constructor defines a string with minimum length of 10. One can use Maybe constructor to determine that a field may not be present in the data being passed. The type with the applied constructors is the new type. There are more type constructors defined by the language.

An泽 language type system supports subtype polymorphism. Let us assume there is a function waiting for input data of type A, but it is transmitted data B. If type A and provided type B are actually different, and the function will handle B correctly, it can be concluded that B can be subtype or type equivalent to the type A.

For example, listing 2 describes type A, which contains a string type field named f1. It also describes type B, which contains the same string field f1 and, optionally, an integer field f2. Since type B contains the same fields of the same type as A, we can say that A <: B.

```haskell
Listing 2. A and B subtype
type A = {
    f1 :: String
}  
type B = {
    f1 :: String
    f2 :: Integer
}
```

That is, B is subtype of A if every term B can be safely used in the context where A is expected.

\[ \Gamma \vdash x : A \quad A <: B \quad \Gamma \vdash x : B \]

(1)

Inheritance is reflective: A <: A and transitive. For instance, listing 3 shows an example of A, B and C, for which the following equation is true: A <: B, B <: C and A <: C.

```haskell
Listing 3. Transitivity
type A = {
    f1 :: String
}  
type B = {
    f1 :: String
    f2 :: String
}  
type C = {
    f1 :: String
    f2 :: String
    f3 :: String
}
```

Moreover, depth subtyping is true, i.e. the types of each corresponding field of a composite type may vary, but should be in terms of inheritance, as in the listings 4 and 5.

```haskell
Listing 4. Inheritance in depth
type A = {
    f1 :: String
}  
type B = {
    f1 :: MinLength 10 String
}
```

```haskell
Listing 5. Inheritance in depth
type A = {
    f1 :: {
        sf1 :: String
    }
}  
type B = {
    f1 :: {
        sf1 :: String
    }
```

15
Rearranging of fields in the description of the user-defined type does not affect subtyping.

If there are two types \( A \) and \( B \) for which \( A <: B \) and \( B <: A \) are true, such types should be considered equivalent.

It should be noted that the application of some constructors to any type forms its subtype. For example, let \( A \) be a base string type, that is \( \text{type } A = \text{String} \), and type \( B \) be a base string type with the applied string maximum length constraint \( \text{constructor type } A = \text{MaxLength 10 String} \).
Then, \( A <: B \) is true. B. Language defines constructors \( \text{List, Maybe and Either} \), which do not form a subtype due to their higher complexity.

Applying different constructors or the same constructor with a different parameter to any type leads to the appearance of two new different types. For example, \( \text{type } A = \text{MaxLength 10 String} \) and \( \text{type } A = \text{MaxLength 20 String} \) can not be considered in terms of inheritance or equivalence.

B. Functions

The types in Anzer are used to describe function’s arguments and results of its operation. A function is either a reference to a repository with its source code or a synonym for the composition of other functions. Therefore, Anzer does not provide writing application logic, making it possible only the type-safe composition of functions having been implemented in other languages. An example of the system description is shown in the listing 6.

Listing 6. An example of the system description in Anzer

```haskell
type RawAddress = MinLength 10 String

type Address = {
    street :: Maybe MinLength 10 String
    city :: MaxLength 20 String
    country :: String
}

github.com/u/parse[go]::
    RawAddress -> Address
isp github.com/u/isprovider[go]::
    Address -> Bool

detect :: RawAddress -> Bool

detect = isp . parse

detect = detect,
```

C. Error handling

To handle errors you can use the type constructor \( \text{Either a b = Left a | Right b} \) which specifies that the function returns data of either type \( a \) or type \( b \). For example, \( \text{Either Error Result} \) defines an algebraic data type which means that the result can be either an \( \text{Error} \) type or the \( \text{Result} \) type. There is no predefined error type.

The composition of functions which returns the result \( \text{Either Error Result} \) with functions expecting only the \( \text{Result} \) type is performed using \( \text{Either} \) monad. Its definition and use are similar to that of the Haskell programming language [8]. Construction \( \text{Either} \) in Anzer defines 2 operations: \( >>= \) and \( \text{return} \).

Operation \( >>= \) (bind) is the higher-order function of Anzer, takes some data and another function as arguments and is defined as follows:

```
Right a >>= f = f a
Left a >>= f = Left a
```

Given example shows that if the first function argument \( >>= \) is of type \( \text{Right a} \), where \( a \) is a user-defined data type, then \( >>= \) converts data of type \( \text{Right a} \) into type \( a \). Then, the function \( f \) passed by the second argument is applied to the given data, and the result of this application is the result of the operation \( >>= \).

If the first binding argument is of \( \text{Left a} \) type, the operation returns the data passed to it unchanged.

Thus, using the \( >>= \) operation it is possible to bind functions returning an error message instead of operation result with functions expecting only correct data, not an error. At the same time, once generated, the error will reach composition’s end without changes.

Another operation of \( \text{Either} \) monad is \( \text{return} \) function, which can be defined as follows: \( \text{return a} = \text{Right a} \). As you can see from the definition, \( \text{return} \) casts user-defined data of type \( a \) to type \( \text{Right a} \) by means of the \( \text{Right} \) constructor. With the help of this operation you can bind functions that return type \( a \) instead of \( \text{Right a} \).

Since Anzer is the domain-specific language, not a general-purpose language, unlike Haskell or other similar functional languages, there is no provision for creating custom type constructors, monads, or higher-order functions.

IV. ANZER PLATFORM

One composition language would not be enough to achieve the goal, so the proposed solution also includes a specialized platform, the tasks of which include:

- Type checking in the description of the function composition in Anzer.
- Functions building using a specialized library to ensure compliance of the declared and implemented types.
• Deployment of new versions of functions in the selected FaaS platform.

A. General organization

To ensure compliance of the declared and implemented types, a specialized library is used that encapsulates all interaction with the FaaS platform. In addition, the proposed solution has a built-in code generator that prepares the basic structures or interfaces based on the data types described in the Anzer language. After generating a function basis, developer only needs to implement the business logic of an application. When you deploy a function in the platform, generating additional function code based on the description in the Anzer language comes first. Additional code is required to confirm that the function implementation matches the description. If the implementation does not match the description declared in the Anzer file, the implementation language compiler would report an error.

The schematic representation of the system consisting of Anzer, the FaaS platform and functions is shown in figure 1.

The square with the caption “Anzer-λ” in the scheme shows a component implementing the logic of the application directly. This part is implemented by developer by means of the selected programming language. Arguments for launching in the function’s code and the return of the work result are available using the Anzer library. The library, in turn, interacts with the selected FaaS platform.

The square with the caption “λ” directly shows a container with an executable file launched by the FaaS platform. The data bus is the part of the FaaS platform.

The rectangle labelled “Anzer Platform” indicates that the functions are managed through the appropriate user interface. Despite this, it remains possible to use standard tools of the selected FaaS platform.

The typical process of creating a new system in the FaaS platform using the proposed solution is as follows:

1) Describing required data types.
2) Generating a necessary function’s basis in terms of the described types by means of Anzer user interface.
3) Implementing the function’s operating logic.
4) Deploying a function in the version control system and describing functions in Anzer according to the same scheme as that of types.
5) Deploying the implemented functions via the Anzer user interface. What happens alongside:
   a) Verification of function composition availability based on the specified types.
   b) Based on the described types generation of missing code to work within the target FaaS platform.
   c) Compilation (if possible) of the implemented function and the code generated in step 5.b.
   d) In case of the successful container’s compilation here comes the deployment of the functions in the FaaS platform.

The languages for which there is a compiler or static analyser with the possibility of static type checking in the code can be used as the languages of the function implementation. Such languages can be, for example, C++, Go, Java, TypeScript or PHP.

B. User interface

The user interface of the solution is a set of utilities with CLI (Command line interface) interface. The utilities use a document in Anzer language to generate the basis of a function, to build it and to deploy it in the FaaS platform.

For instance, calling `anzer g -i scheme.anz -o ~/go/src/a/a.go -f parse` generates a basis for the function `parse` which is shown in the listing 6.

Calling `anzercli build --anz listing2.anz parse` builds a container from source code of the `parse` function.

V. IMPLEMENTATION AND TESTING

A. Implementation

Project Apache OpenWhisk [4] has been selected as the first supported FaaS platform due to ease of deployment, adequate performance [10] and the availability of the required functionality to implement the proposed solutions [9]. The proposed solution is implemented by means of Golang programming language which was chosen due to having necessary competence. Having Golang supported by main suppliers of serverless cloud services (AWS Lambda Functions Azure, Google CloudFunctions) enabled to use it as the first language supported by the Anzer platform.

The developed software includes:
• CLI user interface;
• Anzer parser and interpreter;
• A component to work with OpenWhisk:
  – A client for the OpenWhisk HTTP API;
  – A library for Golang that encapsulates the work with the FaaS platform.

Package of functions, i.e. a collection of bound functions and triggers, is created for each document in Anzer language with the help of the OpenWhisk client. A trigger, by means of
which a function can be called, is created for each function. In the case of function composition, the code required to generate an event for calling the next function is generated automatically. Rules for calling functions for HTTP events, database events etc. can be configured by OpenWhisk built-in tools.

A partial example of a template that could be generated by the command from Chapter IV-B is shown in the listing 7.

Listing 7. Generated Go-template

```golang
    type TypeIn struct {
        Price float64 `json:"price"
        Text string `json:"text"
    }
    type TypeOut struct {
        Desc string `json:"desc"
        Name string `json:"name"
        Price float64 `json:"price"
    }
    func Handle(input TypeIn) TypeOut {
        var out TypeOut
        return out
    }
```

It is notable that there is no line length check in the given sample code. The specified data type requirement is checked by the Anzer interpreter, but checking in the function code is yet to be implemented. After generating the template, you should implement business logic in the `Handle` function.

The code required to work in OpenWhisk will be generated at the stage of function deployment. Due to the use of interfaces in it, it is impossible to change the format of the transmitted data without editing the Anzer document.

B. Comparison with analogues

Comparison with analogues was made using criteria from [5]:

1) ST-safeness [11]: the solution meet the criterion as:
   - The function composition is a new function.
   - The composition of functions does not incur additional costs of computing power and, as a consequence, financial costs.
   - In the proposed solution, the following function is called asynchronously, that is, the execution time of all functions in the composition is not summed.

2) Programming model: functional-like DSL.

3) Parallel execution support: not supported at the moment.

4) State management: the presented solution uses the Open-Whisk platform that, in turn, uses the Apache Kafka message broker in the data transmission channel between functions. Therefore, the maximum size of the transmitted state is equal to the maximum size of the message in Apache Kafka.

5) Software packaging and repositories: the source code of functions is stored in Git repositories.

6) Architecture: uses an OpenWhisk architecture consisting of a controller and a message queue.

7) Overhead: absent as Anzer is not a component of the FaaS platform and responsible only for configuring the interaction between functions.

8) Billing model: not applicable.

There is no type safety in the list of criteria because no alternative solutions matching this criterion have been found.

According to the criteria of the article [5], the proposed solution is not inferior to the alternative. However, if we take into account possibilities presented by Anzer, it is less functional, which means it is impossible to use it to build complex systems yet.

It should also be noted that the Anzer platform does not increase the system’s consumption of machine resources and the same goes for the operating time of functions, as in fact it only adjusts the connection between them.

C. Testing

The proposed solution is being in the process of testing. Currently, Anzer is used in several simple systems, one of which is used to simplify the process of passing code-review. The system consists of 3 functions. Two functions perform specific actions and return a result by a repository management system event. The third one receives an action result and sends a text message to a chat.

An example of the scheme used in testing is shown in the listing 8. The `Hook` type describes a small part of the query that is automatically sent by the project management system Gitlab when certain events occur. The `Event` type contains validated information about edits in the code in the repository. The `Assignment` type is used to transfer information about the person assigned to control the current edits in the code.

The `validate` function, in accordance with its name, checks the incoming query and, using Gitlab API, determines the programming language used in the repository. The `assign` function appoints the person responsible for checking code edits based on the programming language of the repository. The `notify` function sends an appointment notification to the chat.

Listing 8. The example of using Anzer

```golang
    type Hook = {
        username :: String
        repository :: {
            name :: String
            homepage :: String
        }
    }
    type Event = {
        author :: String
        repository :: String
        language :: String
    }
    type Assignment = {
        reviewer :: String
        repository :: String
    }
```
The given example is the very simple system not using even a possibility to handle errors. Nevertheless, its construction made it possible to verify the viability of the approach at the minimum level.

To apply the platform and Anzer language to the real-world problems, for example, building ETL systems, a number of language and platform improvements are required, for instance:

1) The support of the conditional operator present in alternative means of composition is required. In the Anzer language, it could be a pattern-matching analogue from functional programming languages.

2) Using the project as a tool to create full-fledged commercial products requires the development of specialized tools for function debugging and testing.

In general, the use of Anzer together with the accompanying software facilitates creation of new systems and support of existing ones, while not affecting the system requirements for hardware resources and performance.

VI. CONCLUSION

The article considers the problem of function interaction interface divergence in the FaaS platforms. When you develop a system using this approach, you may experience a situation where some components wait or return data in an updated format that is incompatible with outdated components. One of the reasons for this problem is the lack of tools to describe and verify the type checking of interacting functions.

As a solution, this article proposes domain-specific language Anzer created to easily describe the types of data transmitted, the type-safe composition of functions within the whole system, the semantics of their interaction and the logic of error handling. The developed software enables to automate the creation of new functions and maintain their integration with others at all stages of system development and support. Together language and software prevent errors due to mismatch between input data format expected by a function and actually transmitted one. All this makes it possible to simplify and speed up the process of developing systems based on the concept of Function as a Service.

The extensive use of the type system in describing the functions interaction distinguishes the proposed solution from the alternatives, but there are several unresolved problems:

1) A possibility to select a function for a composition based on the actually transmitted data type is needed. In the current implementation, in case of using an algebraic type function, for instance, Left a | Right b, the final function is obliged to process both variants. For full use it is necessary to implement an analogue of the pattern matching operation in functional languages.

2) In addition to pattern matching, “if-then” construction working with data will simplify the description of complex interaction schemes.

3) Solving problems that are more complex than those described in the “Testing” section requires the possibility of local functions testing and debugging.

An important point is to develop the possibility of using Anzer language along with one of the cloud FaaS platforms, such as IBM CloudFunctions, AWS Lambda or others, as it is them whom the most mature alternative composition solutions are created for.

REFERENCES

Path planning for UAV search using growing area algorithm and clustering

Alexander Fedorov
Higher School of Economics
Saint Petersburg, Russia
afedorov2602@gmail.com

Abstract—This paper presents a novel approach to planning a path for unmanned aerial vehicles (UAVs) for searching lost targets under uncertainty. A new basic approach for fast and effective solutions is developed which outperforms standard local hill climbing (LHC) approach. This can be improved using a clusterization of probabilities in the area. The idea allows us to simplify the problem of planning a UAV path to a problem of choosing several clusters. Finally, the proposed method is compared with other methods in simulated scenarios. The comparison shows its high efficiency to solve searching problems.

Index Terms—unmanned aerial vehicles (UAVs), path planning, object detection, probability distribution

I. INTRODUCTION

Unmanned aerial vehicles are gaining popularity both in civilian and military fields. They can be used for wildlife monitoring, target tracking, reconnaissance, surveillance, search and rescue, area patrolling and battle damage assessment. For many of these mission time is critical and should be minimized. Minimizing time in this paper means minimizing length of UAV paths.

Most of the problems mentioned above cannot be solved optimally in polynomial time. The complexity of a general searching problem is NP-hard [1]. This paper focuses on effective solutions which means that the proposed method can be only a suboptimal heuristic polynomial-time solution.

II. PROBLEM FORMULATION

There are various types of UAVs with different characteristics, but we will have the following assumptions:

- UAVs can maintain a constant height above the ground;
- UAVs have a gimbaled camera;
- at some constant height, UAVs see below using the camera a square of size $a$ on the ground;
- UAVs travel with known constant speed;
- UAVs can make a 90-degree turn if necessary.

In the paper, we supposed that the area of search is discretized into $N = W \times H$ squares of size $r$ which means that flying above the center of a square at known constant height a UAV can see it fully. For each square $(i, j)$ (denoted by the number of its column and row) the probability that the target is in this square is known and equal to $p_{i,j} \in [0, 1]$. See Fig. 1 for the example of such discretization with known probabilities.

A typical size of a probability grid for a search problem is tens of thousands squares [3]. The total probability in the search area should be equal to 1 and thus

$$\sum_{i,j} p_{i,j} = 1. \quad (1)$$

We denote a UAV path as $R = (R_1, \ldots, R_k)$, where $R_i \in \{1, \ldots, W\} \times \{1, \ldots, H\}$, the initial cell where the drone is located as $S \in \{1, \ldots, W\} \times \{1, \ldots, H\}$ and the length of the path as $L$.

Then the search problem then can be formulated as follows: Given probability grid with known probabilities $p_{i,j}$, the initial position of an UAV $S$ and maximum length of the path $L$ we should:

$$\max_{R} \sum_{(x,y) \in R} p_{x,y}$$

subject to $R_1 = S$

and $\sum_{i=1}^{k-1} \text{distance}(R_i, R_{i+1}) \leq L,$

where $\text{distance}(a, b)$ is Euclidean distance between centers of cells $a$ and $b$.

In other words, we want to find a path that

- visits some squares and collects probabilities from them by performing a scan;
- has a length no more than $L$;
- maximizes probability collected.

Note that there is no benefit from visiting (scanning) the same square twice.

2. Fig. 1: Example of a probability grid [2]
III. RELATED WORK

This problem is a special case of Orienteering problem [4] with a grid structure instead of a general case of a graph. Orienteering problem is a more complex version of traveling salesman problem (TSP), is known to be NP-hard and has no fully polynomial approximation scheme unless P=NP [5].

The considered problem may be solved with mixed integer linear programming (MILP) [6], genetic algorithm [7] and ant colony optimization [8]. Nevertheless, these approaches are effective only for a small number of nodes (less than 100).

The basic approach for fast solutions of this problem usually is local hill climbing [9]. It is a greedy solution in which in each step a UAV flies to one of neighbours of its current cell choosing the cell with the maximum probability. However, the main problem of this approach is that the drone can’t leave an area of local maximum and fly to a better area unless it covers the area fully (see Fig. 2).

![Fig. 2: Example of a problem of LHC when a drone can’t get to an area of high probability. The more green a cell is the more probability it contains. The route is blue and the initial UAV position is a purple square.](image)

Different authors tried to overcome the problem. Lanillos et al. supposed to calculate not only real probability the drone collects flying in a direction but also expected probability that it can collect there [10]. Moreover, he uses receding horizon optimization (RHC) to increase the number of steps the drone look ahead and to make routes more locally optimal. Yao et al. clusterize an area to detect areas of high probability [11]. In each of the clusters, he builds routes separately using RHC and then joins them. To use clustering the probability distribution is supposed to be good, i.e. we can clearly see subareas of high and low probability. Lin et. al proposes using of global warming effect optimization which allows the drone to ignore cells with low probability [3].

IV. PROPOSED ALGORITHM

A. Issues of analogs

As it was stated before, effective (fast) solutions for the problem are usually based on local hill climbing algorithm. However, as a consequence, these algorithms ([10][11][3]) have the following issues (Fig. 3):

- Final routes visit a lot of cells twice. Visiting a cell more than once doesn’t increase the probability of detecting. It doesn’t always mean that the solution is non-optimal but can be usually avoided in such a way that the resulting route will have higher probability collected;
- A solution may miss cells with high probability because in some moment visiting them is locally non-optimal, but visiting them in future may be valuable. Local hill climbing algorithm is unlikely to visit these cells in future because in each step in chooses the next cell only from four adjacent cells.

![Fig. 3: Example of problems in solutions generated by algorithms based on local hill climbing. In black circles there are cells that are visited twice. In purple circles there are cells with high probability that are skipped by the solution. In the yellow circle there are cells that can be skipped instead of cells in purple circles.](image)

B. Basic idea

**Lemma.** Any area of cells consisting of connected cells of double size (i.e. consisting of four small cells) may be covered with a hamiltonian cycle which goes through centers of small cells.

The proof of the lemma may be found in [12].

The algorithm to get the hamiltonian cycle is simple:

1) Build a spanning tree on cells of double size (Fig. 4). It can be built because the area is connected.
2) A path along the spanning tree is a hamiltonian cycle needed (Fig. 5)

![Fig. 4: (a) Area of cells of double size. (b) A spanning tree on cells of double size.](image)

This lemma brings us to the following idea: Instead of building a path we will build an area of cells of double size.
To maintain connectivity we may add to the area only cells that are neighboring to it. As in local hill climbing, we can use greedy algorithms, i.e. among all possible cells of double size add the one that contains more probability (this can be done effectively by using, e.g., a heap data structure). Note that at every step we preserve connectivity of an area of cells of double size and it is the only constraint needed to use the lemma.

After building the area we will build a hamiltonian cycle using the lemma and it will be the final route of a UAV. Note that as we will build a hamiltonian cycle we know how many cells we need to add to the area not to exceed the constraint on the length of the final route.

C. Improvement with clustering

This idea may be improved by the use of clustering.

Suppose we have detected all clusters (subareas of high probability). If we join centers of all clusters and the starting position of a drone by an area of cells of double size (e.g. using minimal spanning tree (MST) or Steiner tree, Fig. 6) and then run our greedy algorithm then the area will grow accurately in clusters (Fig. 7).

In some cases (when the constraint on the length of the route is tight) it is better to join not all cluster but only a few of them. To choose what clusters to join we may iterate over all possibilities (if there are a small number of clusters, which is a pretty common case) or just try to pick the nearest cluster step by step (increasing complexity of an algorithm in a linear of the number of clusters time).

Then combining two ideas we get the following algorithm:
1) Choose what clusters to join.
2) Join them and the starting position of a UAV with an area of cells of double size.
3) Grow the area step by step while not exceeding the constraint on the route length.
4) Get the hamiltonian cycle from the area using the lemma.

The total complexity of steps 2-4 is $O(L\log N)$ if using a heap to pick the cell with the highest probability effectively. We will call the algorithm "cluster area-growth algorithm".

V. Experiments

The proposed cluster area-growth algorithm was implemented and compared with local hill climbing algorithm and lhc-gw-conv algorithm from [3] (with global warming effect).

The comparison was performed on several test areas. Some of the areas were generated using several Gaussian distributions and some of these areas were obtained from height maps of real locations (because the form of those maps and real probability maps are similar). For every area, there were several measurements with different constraints on the length of the route ($L$). In the cells, there are total probabilities collected by solutions in given constraints on a particular test.

As it can be seen in Table I the proposed algorithm outperformed other algorithms almost in every case. Basic local hill climbing approach gives dramatically worse results in all tests. The proposed solution gained worse result only when constraint on the length of the route is tight. It is because other solutions were not obliged to return the drone to its starting position (i.e. to form a cycle). As a result, their routes got the opportunity to cover more clusters or to cover more distant clusters (see Fig. 8 and Fig. 9). However, as the constraint increases the proposed solution becomes to get better and better results in comparison with other solutions (see Fig. 10 and Fig. 11), because of the absence of those issues mentioned above.
TABLE I: Results of testing

<table>
<thead>
<tr>
<th>Test #1</th>
<th>Test #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>L=3000</td>
<td>L=7000</td>
</tr>
<tr>
<td>proposed</td>
<td>14%</td>
</tr>
<tr>
<td>lhc</td>
<td>13.3%</td>
</tr>
<tr>
<td>lhc-gw-conv</td>
<td>14.1%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test #3</th>
<th>Test #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>L=3000</td>
<td>L=7000</td>
</tr>
<tr>
<td>proposed</td>
<td>19.6%</td>
</tr>
<tr>
<td>lhc</td>
<td>19.2%</td>
</tr>
<tr>
<td>lhc-gw-conv</td>
<td>19.4%</td>
</tr>
</tbody>
</table>

Fig. 8: Test 1 with constraint $L = 3000$, proposed solution

Fig. 9: Test 1 with constraint $L = 3000$, lhc-gw-conv

VI. CONCLUSION

The proposed solution has following next advantages:

- The idea simplifies the problem of building a path over an area to a problem of choosing several clusters (number of clusters is far less than the number of cells in the area);
- The solution can not visit the same cell twice because we have a hamiltonian cycle;
- The solution does not miss probabilities because an area can grow in all directions (not only 4 adjacent cells may be added);
- As a result, it has better efficiency (according to the tests);
- It always returns a drone to the starting position which is good when the constraint on the length of the route is due to limited battery charge of a UAV.

We compared the solution with other solutions in simulated scenarios to show its high efficiency. The ideas from the paper may be also used in other search problems to simplify them and get more practical solutions. In future work this approach can be improved by considering the number of turns in the path (which should be also minimized).
ACKNOWLEDGEMENTS

This paper was done under research work in Simlabs Inc. (sim-labs.com)

REFERENCES


Machine Learning Methods for Earthquake Prediction: a Survey

Alyona Galkina  
Saint Petersburg State University  
Saint Petersburg, Russia  
id.a.brickman@gmail.com

Natalia Grafeeva  
Saint Petersburg State University  
Saint Petersburg, Russia  
n.grafeeva@spbu.ru

Abstract — Earthquakes are one of the most dangerous natural disasters, primarily due to the fact that they often occur without an explicit warning, leaving no time to react. This fact makes the problem of earthquake prediction extremely important for the safety of humankind. Despite the continuing interest in this topic from the scientific community, there is no consensus as to whether it is possible to find the solution with sufficient accuracy. However, successful application of machine learning techniques to different fields of research indicates that it would be possible to use them to make more accurate short-term forecasts.

This paper reviews recent publications where application of various machine learning based approaches to earthquake prediction was studied. The aim is to systematize the methods used and analyze the main trends in making predictions. We believe that this research will be useful and encouraging for both earthquake scientists and beginner researchers in this field.

Keywords — earthquake prediction, data mining, time series, neural networks, seismology

I. INTRODUCTION

At present, many processes and phenomena affecting different areas of human life have been studied enough to make predictions. Risk analysis makes it possible to determine whether the event is likely to occur at given period of time, as well as promptly respond to this event or even prevent it. However, even in the modern world there are events that we cannot influence. Such events, in particular, include natural disasters: tsunamis, tornadoes, floods, volcanic eruptions, etc. Human beings cannot stop the impending threat; but precautionary measures and rapid response are potentially able to minimize the economical and human losses.

However, not all natural disasters are equally well studied and “predictable”. Earthquakes are one of the most dangerous and destructive catastrophes. Firstly, they often occur without explicit warning and therefore do not leave enough time for people to take measures. In addition, the situation is compounded by the fact that earthquakes often lead to other natural hazards such as tsunamis, snowslips and landslides. They may even cause industrial disasters (for instance, Fukushima Daiichi nuclear disaster was initiated by the Tōhoku earthquake that occurred near Honshu Island on 11 March 2011 and was the most powerful earthquake ever recorded in Japan [1]).

All these facts make the problem of earthquake prediction critical to human security. Since the end of XIX century, researchers in seismology and related branches of science have tried to discover so-called precursors, anomalous phenomena that occur before seismic events. Many possible precursors have been studied, including foreshocks (quakes which occur before larger seismic events), electromagnetic anomalies called “earthquake lights”, changes of groundwater levels and even unusual animal behaviour. In some cases precursor appearance led to timely evacuation of civilians [2].

It is important to note that it is hard to use precursors for short-term forecasting, as they are they are not only characteristic of earthquakes (for instance, unusual lights in atmosphere may appear before geomagnetic storms or have a technogenic origin). Furthermore, different precursors preceded the quakes, which had different nature, occurred in different seismic zones and even seasons.

Thus, optimistic attitude towards the possibility of timely detection of earthquake hazards, which emerged in the 1970s because of a number of successful “predictions”, have been replaced by skepticism [3]. This happened primarily because of numerous high-profile cases of wrong predictions [4]. Another reason was that no statistically significant precursors were found [5].

Currently there is no general methodology for earthquake prediction. Moreover, there is still no consensus in science community on whether it is possible to find a solution of this problem. However, rapid development of machine learning methods and successful application of these methods to various kinds of problems indicates that these technologies could help to extract hidden patterns and make accurate predictions.

These tendencies fully explain the amount of papers where the applicability of various machine learning algorithms to the tasks of earthquake science is studied. Some of them are focused on precursor study: for instance, in paper [6] random forest algorithm is applied to acoustic time series data emitted from laboratory faults in order to estimate the time remaining before the next “artificial earthquake”. Another application is discovering patterns of aftershocks which are small quakes that follow a large earthquake (referred to as a mainshock) and occur in the same area. One of the most recent examples is paper [7], where an artificial neural network in trained on more than 130,000 mainshock-aftershock pairs in order to model aftershock distribution and outperforms the classic approach to this task. However, although these fields of research are both very interesting and potentially helpful for solving the problem of earthquake prediction, the task formulated in the papers differs from the original one defined by seismologists (the definition is given in Section II), and therefore the results of these studies cannot be fully compared with the others.

However, despite the undoubted relevance of the problem, the whole time the research have been conducted, only a few authors have tried to systematize knowledge from various sources. In particular, one recent survey on a similar topic was found, published in CRORR Journal in 2016 [8]. The paper reviews using artificial neural networks for short-term earthquake forecasting. However, it is focused only on a single aspect of the problem: the authors mostly discussed various architectures and topologies of neural network models used to solve the problem. Therefore, the paper refers mainly to a limited group of specialists. The main objective of our review
is, on the contrary, to try to narrow the gap between seismology and computer science, as well as to encourage further research in this area. That is why this paper will attempt to cover all the main parts of a process of making predictions, including the search and preprocessing of earthquake data, the principles of feature extraction, as well as the methods of assessing the performance of machine-learning based predictors.

II. DESCRIPTION OF THE TASK

Despite words “forecast” and “prediction” are often used interchangeably, in earthquake science it is customary to distinguish them. Particularly, in [9] the idea was expressed that an earthquake prediction implies greater probability than an earthquake forecast; in other words, a prediction is more definite than a forecast, it requires greater accuracy. Therefore, it is worth noting that in this study we will deal mainly with earthquake prediction, since it seems to be more important from a practical point of view.

According to [10], the following information is required from the prediction of an earthquake in its simplest interpretation:

- a specific location;
- a specific time interval;
- a specific magnitude range.

Importantly, all of these parameters should be defined in such a way that one could objectively state that some future earthquake does or does not satisfy the prediction. It is necessary for both using and evaluating predictions. In particular, it is required to define “location” clearly and determine the exact spatial boundaries of the area, since an earthquake does not occur at a point.

Besides, the prediction is more useful and statistically verifiable if it includes the probability that the event which meets all above-mentioned criteria will occur [11]. That is, a prediction should specify where, when, how big the predicted earthquake is, and how probable it is that it will occur in actual.

However, despite the importance of the problem of earthquake prediction and the existence of precise criteria that its solution should satisfy, there is still no general method for predicting earthquakes with sufficient accuracy. One of the main reasons is that it is extremely hard to build an accurate model of the process of earthquake occurrence. That is due to several reasons:

- Not all factors that may play roles in earthquake occurrence are discovered;
- Even well-known factors, such as the accumulated stress or seismic energy release rate, cannot be directly measured (or it is too hard to do it);
- The relationships between the occurrence of new earthquakes and these seismic features are shown to be complicated and highly non-linear.

All this leads to the use of increasingly complex methodologies when trying to model earthquakes. Some of them will be described below.

III. DATASETS

When a specific field is researched in terms of machine learning, the first question is where to find data. As for earthquake datasets, various organizations and research institutions are constantly monitoring seismic activity of all over the world. There are some open-source national databases and earthquake catalogs, such as seismicity catalogs of Seismological Institute, National Observatory of Athens (http://www.gein.noa.gr/en/seismicity/earthquake-catalogs, Greece), “Earthquakes of Russia” database of Geological Survey, Russian Academy of Sciences (http://eqru.gsras.ru/, Russia), earthquake list of National Institute of Geophysics and Volcanology (http://cnt.rm.ingv.it/en, Italy) et al. There are also public earthquake catalogs provided by international organizations, which contain earthquake data from all over the world. Some examples are USGS catalog (https://earthquake.usgs.gov/earthquakes/search/), EMSC earthquake database (https://www.emsc-csem.org/) and ANSS Composite catalog by Northern California Earthquake Data Center (http://www.ncedc.org/anss/).

Speaking about the structure of earthquake data, it is usually presented in the form of a table, each record of which corresponds to a certain seismic event. The sets of attributes are different for data published in different catalogs, but the most common ones are:

- time of an event’s occurrence;
- geographical coordinates of an epicenter;
- depth of a hypocenter;
- magnitude value, which characterizes the overall “size” of an event and is obtained from measurements of seismic waves recorded by a seismograph;
- magnitude scale used when computing the magnitude value. Several scales have been defined, some of which are easier to compute but have limited applicability, as they cannot satisfactorily measure the strength of the largest events. However, all commonly used scales yield approximately the same values for any given seismic event.

It should be noted that the number of records in all public databases is also different for different countries. It depends not only of seismic activity, but also of development of earthquake monitoring systems in these regions. For example, Japan is known to be the country with the biggest amount of earthquakes recorded. However, according to USGS, the most seismically active place in the world is Indonesia, and Japan has the densest seismic network, which helps them to record more earthquakes [12].

Different level of completeness of earthquake catalogs leads researchers to the need to assess the quality of data they have. There are many different methods of evaluation, one of which is based on Gutenberg-Richter’s law [13] – an empirical law that describes the relationship between earthquake magnitude (M) and frequency of occurrence of events (N) for a given region and a time range. It is expressed as:

$$\log_{10} M = a - bN;$$  \hspace{1cm} (1)

i.e. the frequency rises exponentially with decreasing magnitude. This relation is remarkably resistant in space and time, so data from complete catalogs should also correspond
to it. The study is performed by plotting magnitude distribution. The point where curve starts to deviate from exponential behavior is chosen as a so-called «cut-off magnitude» (it is stated that some events of a magnitude below this threshold value are missing, so the catalog is incomplete). Therefore, the events of magnitude lower than the cut-off value are removed from the dataset. The illustrations of Gutenberg-Richter law for some frequently studied seismic zones are given in Fig. 1, 2 and 3.

IV. PERFORMANCE MEASURES

In this section, the definition is given for the performance measures that are used in literature to evaluate the prediction models.

The simplest metrics used for quality assessment are:

- **True Positive** (TP): The number of outcomes where the model predicted an earthquake and it actually occurred.
- **False Positive** (FP): The number of outcomes where an earthquake was predicted but did not occur in actual.
- **True Negative** (TN): The number of outcomes where the model predicted no earthquake and there was no earthquake in actual.
- **False Negative** (FN): The number of outcomes where the model predicted no earthquake but it actually occurred.

These measures are summarized in a so-called confusion matrix where all possible outcomes are depicted:

<table>
<thead>
<tr>
<th></th>
<th>Occurred</th>
<th>Didn’t occur</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Was predicted</strong></td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td><strong>Wasn’t predicted</strong></td>
<td>FN</td>
<td>TN</td>
</tr>
</tbody>
</table>

There are also some other common criteria derived from the above-mentioned ones. Two of the most common statistical measures are sensitivity (denoted by $S_n$, also called $POD$, that stands for probability of detection), or a rate of actual positives predicted correctly, and specificity (denoted by $S_p$), or a rate of actual negatives predicted. They are defined as shown in Eq. 2 and Eq. 3, respectively:

$$S_n = \frac{TP}{TP+FN};$$

$$S_p = \frac{TN}{TN+FP};$$

Two other important evaluation measures are $P_1$, or a positive predictive value, and $P_0$, or a negative predictive value. $P_1$ is the percentage of true positives out of all positive predictions, as mentioned in Eq. 4, and $P_0$ is the percentage of true negatives out of all negative predictions of a model, as shown by Eq. 5:

$$P_1 = \frac{TP}{TP+FP};$$

$$P_0 = \frac{TN}{TN+FN};$$

Some observed papers also define the criteria called false alarm ratio (denoted by FAR) and frequency bias (FB), as shown by Eq. 6 and Eq. 7, respectively:

$$FAR = \frac{FP}{TP+FP} = 1 - P_1;$$

$$FB = \frac{TP+FP}{TP+FN};$$

Accuracy is also computed from four elements of the confusion matrix. It indicates the percentage of number of accurate predictions out of all predictions made by the model. Accuracy is defined as follows:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN};$$

When earthquake prediction problem is formulated as a binary classification task, another performance criteria used are $R$ score and Matthew’s correlation score (denoted by MCC). They are proposed as balanced evaluation measures and are defined as shown in Eq. 9 and Eq. 10, respectively:

$$R = \frac{TP+TN-FP-FN}{(TP+FN)(FP+TN)},$$

$$MCC = \frac{(TP\cdot TN - FP\cdot FN)}{\sqrt{(TP+FN)(TP+FP)(TN+FP)(TN+FN)}}.$$
\[ MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \] (10)

Finally, in some papers where regression approach is applied to earthquake prediction, such standard measures as mean absolute (MAE) and relative errors (RE) are used. They are computed as follows:

\[ MAE = \frac{1}{n} \sum_{i=1}^{n} |\hat{y}_i - y_i|; \quad (11) \]
\[ RE = \frac{1}{n \max(\{y_i\})} \sum_{i=1}^{n} |\hat{y}_i - y_i| \] (12)

V. REVIEW OF EXISTING APPROACHES

This section reviews a number of publications where application of machine learning methods to the task of earthquake prediction on various temporal and spatial intervals have been studied. Due to the fact that, as mentioned above, the processes of earthquake occurrence are considered to be stochastic and non-linear, most recent researches in this area are devoted to the applicability of neural networks to this problem. Another machine learning techniques, specifically, various regression and classification algorithms are also reviewed.


Reference [14] was one of the first in proposing artificial neural networks (ANN) for earthquake forecasting. The author, E.I. Alves, was inspired by successful application of similar approaches to the tasks of financial forecasting, which, as he thought, are similar to seismic activity in terms of the chaotic nature of both systems. Financial oscillators such as moving averages (MA), moving averages convergence-divergence (MACD), relative strength index (RSI), etc. were used as input data. The forecast was to indicate time and geographical coordinates of an earthquake within spatial and temporal windows, as well as intensity range on Modified Mercalli Intensity scale (denoted by MMI [15]). The proposed method was tested on data of the region of Azores, Portugal. E. I. Alves stated that it forecasted earthquakes correctly in July 1998 (MMI = 8) and in January 2004 (MMI = 5). However, no statistical measures were computed, so we cannot evaluate the performance of this approach objectively. Though time windows were too wide (the month of the seismic event was forecasted to within ± 5 months), the results were “encouraging” and demonstrated the potential of using neural networks to predict earthquakes.


H. Adeli and A. Panakkat, the authors of [16] and [17], formulate the problem of earthquake prediction as a classification task where the magnitude ranges of the largest seismic event in a pre-defined time window (for instance, 1 month) are the output classes. So, the proposed methods are used to predict the magnitude of the biggest earthquake (within 0.5) in a pre-defined region in the following month.

Reference [16], published in International Neural Systems in 2007, defines eight so-called seismicity indicators – mathematically computed features, which can be used to evaluate the seismic potential of a region. These parameters are based of two models of magnitude temporal distribution. The first one is Gutenberg-Richter inverse power law that was described in section “Datasets”. Another one is characteristic model, which is based on the fact that some seismic zones exhibit periodic trends in release of seismic energy through large earthquakes. Due to the importance of these indicators for the formation of an approach to the study of the subject of earthquake prediction, their description is provided in Table II.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Mathematical expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_\theta )</td>
<td>Elapsed time</td>
<td>( T = t_n - t_1 )</td>
</tr>
<tr>
<td>( M_{\text{mean}} )</td>
<td>Mean magnitude</td>
<td>( M_{\text{mean}} = \frac{\sum_{i=1}^{n} M_i}{n} )</td>
</tr>
<tr>
<td>( dE^{1/2} )</td>
<td>The rate of square root of seismic energy</td>
<td>( dE^{1/2} = \frac{\sum_{i=1}^{n} \log N_i - \sum_{i=1}^{n} \log M_i}{2} )</td>
</tr>
<tr>
<td>( \beta )</td>
<td>b-value (the slope of Gutenberg-Richter curve)</td>
<td>( \beta = \frac{n \sum (M_i \log N_i) - \sum M_i \bar{N}_i}{(\sum M_i)^2 - \sum M_i^2} )</td>
</tr>
<tr>
<td>( \eta )</td>
<td>Mean square deviation</td>
<td>( \eta = \frac{\sum \log_{10} N_i - (a - b M_i)^2}{n-1} )</td>
</tr>
<tr>
<td>( \Delta M )</td>
<td>Magnitude deficit</td>
<td>( \Delta M = M_{\text{max observed}} - M_{\text{max expected}} )</td>
</tr>
<tr>
<td>( \mu )</td>
<td>Mean time between characteristic events</td>
<td>( \mu = \sum (t_i \text{characteristic}) / n_{\text{characteristic}} )</td>
</tr>
<tr>
<td>( c )</td>
<td>Coefficient of variation</td>
<td>( c = \frac{\text{STD of the observed times}}{\mu} )</td>
</tr>
</tbody>
</table>

The logical consequence was paper [17] published in Neural Networks in 2009. In this paper, the authors proposed the architecture of a probabilistic neural network (PNN) as a solution for the same problem that was formulated in [16]. Adeli and Panakkat also used the same set of seismicity indicators as input data for training the network. The model was tested on data for South California seismic zone (33.8-35.4 N° and 114.75-119.25 W°) and yielded good prediction accuracies for events of magnitude 4.5 to 6.0 (R score values between 0.62 and 0.78). However, PNN did not perform satisfactorily for quakes of magnitudes greater than 6.0, yielding R scores in range from 0.0 to 0.5.

Thus, studies [16] and [17] complement each other: the authors propose using RNN for predicting earthquakes of large magnitude, while PNN may be used for small and moderate earthquakes. The researches of Adeli and Panakkat have laid the foundation for a scientific approach to assessing the potential seismic hazard for different regions: the set of eight seismicity indicators proposed by them was used in various studies by researchers from all over the world.

C. J. Reyes et al.(2013)

In paper [18], published in Applied Soft Computing in 2013, another method for earthquake prediction using ANN is proposed. The system is designed to provide two kinds of predictions: a) the probability that an earthquake larger that a threshold magnitude happens in five days and b) the probability that a seismic event within a pre-defined magnitude range might occur. The input for the proposed predictor was based on b-value from Gutenberg-Richter’s law (defined in Table 2); moreover, new seismic parameters were
firstly defined. These parameters are based on Bath’s law [19] and Omori-Utsu’s law [20], which describe the relations between main shock and aftershocks, based on their magnitude and frequency of occurrence, respectively. Four seismic regions of Chile were analyzed: Talca (35-36 S° and 71-72 W°), Santiago (33-34 S° and 71-72 W°), Pichilemu (34-34.5 S° and 72-72.5 W°) and Valparaiso (32.5-33.5 S° and 71-72 W°). A different feed-forward backpropagation ANN was applied to each area, though they all shared the same architecture. The prototype predicted an earthquake each time when predicted probability was higher than a pre-defined threshold value (the thresholds were adjusted to reduce the number of false alarms). Evaluation of proposed methods was conducted using performance measures computed from TP, TN, FP and FN. Comparative analysis was performed using standard methods of classification such as K nearest neighbors (KNN), support vector machines (SVM) and classification via K-means clustering. Despite the individual setting of parameters, the performance of proposed ANN varied greatly depending on the region: the $P_0$ values were 17.4% for Talca, 41.7% for Santiago, 86.7% for Pichilemu and 87% for Valparaiso.

D. G. Cortés et al. (2018)

In study [21], which was published in Computers & Geosciences in 2018, an attempt to predict magnitude of the largest seismic event within the next seven days was made.

The problem of earthquake prediction was treated as a regression task: four regressors (generalized linear models, gradient boosting machines, deep learning and random forest) and ensembles for them were applied. Seismicity indicators proposed by Panakkat & Adeli [16] and Reyes et al. [18] were used as input data. The main feature of the study is that the problem was observed in context of big data analytics: a total 1 GB of data processed by means of a cloud-based information were used for training and testing regression models. In order to evaluate the effectiveness of proposed approaches, mean absolute (MAE) and relative (RE) errors were used as performance measures. Besides, due to the specifics of the task, the time spent on training models was also taken into account. The most effective regressor was random forest (RF), yielding a mean absolute error of 0.74 on average. RF was also one of the fastest, taking only 18 minutes to train the regression models on all data. Particularly, the most accurate predictions of RF were made for moderate earthquakes (magnitudes within a range on $[4, 7]$; MAE<0.26), while regression ensembles performed better on extreme magnitude ranges ($[0, 3]$ and $[7, 8]$). Based on these results, the authors concluded that using more complex regressor ensembles would improve the accuracy of predictions for quakes of large magnitude.

E. M. Moustra et al. (2011)

The main purpose of study [22] published in Expert Systems and Applications in 2011 was to evaluate the accuracy of ANN for earthquake prediction using different inputs. More specifically, the paper highlights two main areas of research. The first case study concerned prediction of the largest seismic event of the following day using only time series earthquake magnitude data, and the second one concerned the use of so-called Seismic Electric Signals (SES) to predict the magnitude of the next seismic event as well as time lag. For the first case, a feed-forward backpropagation neural network was used. An input file contained maximum magnitude value for each day. The model was trained using an earthquake catalog for Greece, and performance was evaluated with accuracy rate, which was calculated based on MAE. The average accuracy rate was 80.55% for all events, but only 52.81% for what Moustra et al. considered “outliers” (earthquakes of magnitude greater than 5.2). In order to improve the performance on major quakes, the authors trained the ANN it two phases (at first on outliers, then on all training dataset), and the resulting accuracy rate was 58.02%.

The case study that concerned earthquake prediction using SES consisted of two major parts. It is noteworthy that at the time of the study only 29 samples of SES were recorded and published by VAN team in Greece. Despite this, the authors of [22] tried using an ANN to study the connection between SES and the occurrence of earthquakes. Due to the fact that 29 samples were clearly not enough to train neural networks, Moustra et al. had decided to construct the missing data for the rest of seismic events from the catalog. In first case, SES were generated randomly for all events; in second one the ANN was used to construct missing data using magnitude time series. The accuracy rate of magnitude prediction was slightly more than 60% on the first dataset, and the ANN found no correlation between SES and the time lag. Using data constructed by the ANN improved the performance significantly: the accuracy rates that resulted from the prediction of both magnitude and time lag were 83.56% for magnitude and 92.96% for time lag. The results have led the authors to conclusion that training models on the appropriate data is a key factor that may influence the resulting performance greatly.

F. K. Asim et al. (2017)

In paper [23], which was published in Natural Hazards in 2017, the problem of earthquake prediction is studied as a binary classification task. Predictions were made for events of magnitude greater than or equal to 5.5 on monthly basis. Eight seismicity indicators proposed by Adeli & Panakkat [16] were used as input to different machine learning classifiers. These included recurrent neural network (RNN), pattern recognition neural network (PRNN), random forest (RF) and LPBoost ensemble of decision trees. In addition to the accuracy of predictions, Asim et al. identified such performance measures as sensitivity and specificity, true and false predictive values as the main criteria for comparison of the above-mentioned approaches. The classifiers were used to predict earthquakes in the Hindukush region. LPBoost ensemble tended to take the lead in accuracy with the value of 65%. This classifier also performed better in terms of sensitivity towards earthquake occurrence, yielding 91% of $S_r$ value. The authors also highlighted the result of PRNN, which produced the least false alarms as evidenced by a high level of positive predictive value equal to 71%. Having analyzed the results, the authors stated that every observed system had shown satisfactory results somehow or other.

G. K. Asim et al. (2018)

An earthquake prediction system (EPS) named EP-GPBoost was described in paper [24], which was published in Soil Dynamics and Earthquake Engineering in 2018. This system is a classifier based on a combination of genetic programming (GP) and a boosting algorithm named AdaBoost. An application of these instruments to the problem of earthquake prediction had never studied before this paper. Another novelty of the approach is a methodology of computation and simultaneous usage of seismicity indicators, which is based on idea of obtaining maximum information...
about geological properties of observed regions (instead of choosing appropriate parameters for each zone individually). A total of 50 features was calculated, based on such geological concepts as Gutenberg-Richter’s law, release of seismic energy, foreshock frequency, etc. Some of these parameters were computed via different approaches (for example, the above-mentioned b-value, which is a slope of a Gutenberg-Richter curve, was computed using two methods, namely, least square regression analysis (as shown in Table 2) and maximum likelihood method). As a result, a system for predicting seismic events of magnitude equal or greater than 5.0 for the next 15 days was proposed. The study of the applicability of EP-GPBoost was performed using data from previously used seismic zones, namely, Chile (32.5–36°S, 70 –72.5°W), Hindukush (35.39°N, 69 –74.6°E) and Southern California (32 –36.5°N, 114.75 –121 W°). The experiments have shown outstanding performance in all three observed regions both in terms of low false alarm ratio (the precision values were 74.3%, 80.2% and 84.2% for Hindukush, Chile and Southern California, respectively) and in terms of other metrics considered for evaluation, such as MCC and R score.

The best results were obtained for the region of South California (the authors stated that the reason was the quality and completeness of the corresponding earthquake catalog). However, the results of all the regions exhibit improvement when compared to the previous studies [16][18][23].

H. K. Asim et al. (2018)

Reference [25], published in PLOS ONE in 2018, was written by the authors of the previous research. In this paper Asim et al. also used the approach to usage of seismicity indicators proposed in [24]. This time, 60 seismic parameters was computed using various concepts of seismology. Again, some specific features were calculated via different approaches to retain the most complete information about the observed seismic zones. As in their previous research, the authors aim to predict the earthquakes of magnitude equal to or greater than 5.0 for the next 15 days. The proposed system is multistep, unlike previous other predictors proposed in literature which are mainly simple. The system is a combination of different machine learning algorithms, and on each step, one algorithm uses the knowledge obtained through learning of a previous one. Firstly, two-step feature selection is used to choose the most relevant parameters for training a model. Specifically, relevance and redundancy checks are performed (Minimum Redundancy Maximum Relevance criteria, denoted as mRMR, is applied). The resulting set of parameters is passed to a support vector regressor (SVR), and the trend predicted by SVR is then used as a part of input data for a hybrid neural network (HNN). A HNN proposed in [25] is a combination of three different ANNs and EPSO algorithm for weight optimization. The resulting system called SVR-HNN was applied to previously studied regions of Hindukush, Chile and Southern California. The performance was evaluated with such measures as $P_0$, $P_1$, $S_n$, $S_p$, accuracy, MCC and R score. The results were also compared with ones described in previous researches on these seismic zones. The resulting values of performance measures (for instance, R score increased from 0.27 to 0.58 for Hindukush, from 0.344 to 0.603 for Chile, 0.623 from 0.5107 to 0.623 for Southern California) showed that proposed multistep methodology improved prediction performance in comparison with individual machine learning techniques.

All reviewed papers are summarized in Table III. An analysis of all the above-mentioned works revealed a number of trends in studying the problem of earthquake prediction. Some of these trends and common approaches are described below.

**VI. DISCUSSION**

This section identifies the main tendencies in earthquake prediction using machine learning techniques and highlights the areas that should be the subjects of further research.

**TABLE III. SUMMARY OF ALL STUDIES REVIEWED**

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Seismic zone</th>
<th>Data sources</th>
<th>ML. methods observed</th>
<th>Features</th>
<th>Evaluation techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>[14]</td>
<td>Azores (Portugal)</td>
<td>-</td>
<td>ANN</td>
<td>Financial oscillators</td>
<td>-</td>
</tr>
<tr>
<td>[16]</td>
<td>South California, San-Francisco bay (USA)</td>
<td>SCEC catalog</td>
<td>LMBP; RNN; RBFN</td>
<td>Seismic indicators based on GR law and characteristic distribution</td>
<td>$S_p$, FAR, FB, R score</td>
</tr>
<tr>
<td>[17]</td>
<td>South California (USA)</td>
<td>SCEC catalog</td>
<td>PNN</td>
<td>-</td>
<td>$S_p$, FAR, R score</td>
</tr>
<tr>
<td>[18]</td>
<td>Chile</td>
<td>University of Chile’s National Service of Seismology</td>
<td>feed-forward ANN</td>
<td>Seismic indicators based on GR law, Bath’s law and Omori-Utsu’s law</td>
<td>$P_0$, $P_1$, $S_n$, $S_p$</td>
</tr>
<tr>
<td>[21]</td>
<td>California (USA)</td>
<td>ANSS Composite Earthquake Catalog; NCEDS</td>
<td>GLM; GBM; DL</td>
<td>Two approaches to modeling SES for all seismic data</td>
<td>MAE, RE</td>
</tr>
<tr>
<td>[22]</td>
<td>Greece</td>
<td>Seismological Institute, National Observatory of Athens (SINOA), VAN team</td>
<td>ANN</td>
<td>Accuracy rate based on MAE</td>
<td>-</td>
</tr>
<tr>
<td>[23]</td>
<td>Hindukush (Pakistan)</td>
<td>Center for Earthquake Studies (Pakistan), USGS</td>
<td>LMBP-RNN; PRNN; RF; LPRBoost</td>
<td>-</td>
<td>$P_0$, $P_1$, $S_n$, $S_p$, accuracy</td>
</tr>
<tr>
<td>[24]</td>
<td>Hindukush (Pakistan), Chile, South California (USA)</td>
<td>USGS catalog</td>
<td>GP, AdaBoost</td>
<td>50 seismic indicators used simultaneously</td>
<td>$P_0$, $P_1$, $S_n$, $S_p$, accuracy, MCC, R score</td>
</tr>
<tr>
<td>[25]</td>
<td>Hindukush (Pakistan), Chile, South California (USA)</td>
<td>USGS catalog</td>
<td>mRMR criteria, SVR, HNN, EPSO</td>
<td>60 seismic indicators used simultaneously</td>
<td>$P_0$, $P_1$, $S_n$, $S_p$, accuracy, MCC, R score</td>
</tr>
</tbody>
</table>
First of all, the definition of an earthquake prediction given by seismologists implies giving the exact definition of time and place of earthquake occurrence as well as its magnitude (as defined in the section “Description of the task”). However, most of the studies observed are focused on wider aim of predicting magnitude for a limited area and temporal range. The summary of temporal, spatial and magnitude limits used in reviewed papers when formulating the problem are given in Table IV.) That is explained by extreme complexity of the process of earthquake occurrence. Further research in this area should be directed towards attempts to simultaneously predict magnitude, time and place of seismic events’ occurrence.

TABLE IV. THE SUMMARY OF ALL THE APPROACHES TO LIMITING THE PREDICTIONS’ ACCURACY USED IN REVIEWED PAPERS

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Time ranges</th>
<th>Geographical boundaries</th>
<th>Magnitude ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>[14]</td>
<td>± 5 months</td>
<td>The latitude of an epicenter is predicted to within 1°</td>
<td>-</td>
</tr>
<tr>
<td>[16]</td>
<td>1 month</td>
<td>Two pre-defined regions: South California (32-36 N°, 114-120 W°); San-Francisco bay (37.5-40 N°, 116-123.5 W°)</td>
<td>The magnitude is predicted to the nearest 0.5</td>
</tr>
<tr>
<td>[17]</td>
<td>15 days</td>
<td>One pre-defined region: South California (33.8-35.4 N°, 114.75-119.25 W°)</td>
<td>The magnitude is predicted to the nearest 0.5</td>
</tr>
<tr>
<td>[18]</td>
<td>5 days</td>
<td>The coordinate ranges varied from 0.5° × 0.5° to 1° × 1°</td>
<td>The probability that the magnitude will exceed the threshold value or will be in a pre-defined range is predicted</td>
</tr>
<tr>
<td>[21]</td>
<td>7 days</td>
<td>The coordinates of an epicenter are predicted to within a cell of 0.5° × 0.5°</td>
<td>-</td>
</tr>
<tr>
<td>[22]</td>
<td>1 days</td>
<td>The pre-defined region of Greece</td>
<td>-</td>
</tr>
<tr>
<td>[23]</td>
<td>1 month</td>
<td>The pre-defined region of Hindukush</td>
<td>The occurrence of events of magnitude ≥ 5.5 is predicted</td>
</tr>
<tr>
<td>[24]</td>
<td>15 days</td>
<td>Southern California (32-36.5 N°, 114.75-121 W°), Chile (32.5-36.5 S°, 70-72.5 W°), Hindukush (35-39 N°, 69–74.6 E°)</td>
<td>The occurrence of events of magnitude ≥ 5.5 is predicted</td>
</tr>
<tr>
<td>[25]</td>
<td>15 days</td>
<td>Southern California (32-36.5 N°, 114.75-121 W°), Chile (32.5-36.5 S°, 70-72.5 W°), Hindukush (35-39 N°, 69–74.6 E°)</td>
<td>The occurrence of events of magnitude ≥ 5.5 is predicted</td>
</tr>
</tbody>
</table>

As for data processing, most of the papers reviewed use the approach of feature extraction based on seismic characteristics of a region. As every seismic zone has its unique parameters, it is obvious that these parameters need to be considered for building an exact model. This “personalized” approach is especially noticeable in some of the studies where various zones were observed: the results show that some approaches performed better on one region and worse on the other. There were also researches where different architectures or even methods were applied for modeling different seismic zones because of their differences. In addition, the principles of feature selection and usage are changing over time: in papers published in 2018 a new approach is proposed, which is based on simultaneous use of a large number of seismic indicators for building and training the predicting models.

It is also noteworthy that a number of researches outlines low false alarm generation as an important criterion of performance evaluation. Many authors indicate that earthquake prediction is a delicate issue where false alarms lead to particularly negative consequences, such as economical losses and panic among the civilians, which can be critical because it may cause distrust of the system. Therefore, in some cases we can even sacrifice the sensitivity of a model in favor of reducing a number of false alarms.

Speaking about the performance of proposed models, it is worth noting that it is hard to compare approaches proposed in different papers, because the researchers use different performance measures for assessing the quality of predictors. That is why one cannot objectively state that one model is better than the other is. However, some conclusions can still be made. First of all, the accuracy of predictions as well as other performance measures increase with the research on the field of earthquake prediction (it is noticeable based on the repeatedly studied regions of Southern California, Chile and Hindukush, where similar performance measures have been used). It is also worth noting that in some papers a tendency is observed concerning the decrease of accuracy with increasing magnitude threshold. That is, the larger the earthquake, the harder it is to predict. Given the fact that large earthquakes represent the greatest threat to society, it is necessary to make bigger efforts in the task of predicting earthquakes of high magnitude (equal to or greater than 5.5).

The models proposed in most of the papers reviewed were tested on data for different regions obtained from different earthquake catalogs. We think that this is a major issue. As shown in a number of papers, an approach may perform differently on zones with different seismic properties, and that is another reason why it is near to impossible to compare the methods proposed in different studies. As a solution, we propose to create a «benchmark» dataset, which researchers can use in comparative purposes for different algorithms. The dataset may contain open-source data on seismic zones used in previous studies, such as Chile, Hindukush and Southern California. Besides, we think that it is necessary to complement the dataset with records from other seismic zones from different parts of the world, for instance, Europe and East Asia. We believe that testing the approaches on unified data from regions with different magnitude distributions and other seismological properties will help to carry out a more detailed study of their applicability. The exact geographical boundaries of regions from the proposed «benchmark» dataset and cut-off magnitudes chosen for these regions based of the study of Gutenberg-Richter curves (as described in section “Datasets”) are listed in Table V. The visualization of seismic activity and magnitude distribution of these regions is shown in Fig 1-5.

TABLE V. RANGES OF COORDINATE BOUNDARIES OF REGIONS PROPOSED AS PARTS OF THE DATASET

<table>
<thead>
<tr>
<th>Seismic zones</th>
<th>Latitude range</th>
<th>Longitude range</th>
<th>Cut-off magnitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Japan</td>
<td>34-39 N°</td>
<td>136.5-142 E°</td>
<td>4.5</td>
</tr>
<tr>
<td>Chile</td>
<td>32.5-36 S°</td>
<td>70-72.5 W°</td>
<td>4.0</td>
</tr>
<tr>
<td>Hindukush</td>
<td>35-39 N°</td>
<td>69-74.6 E°</td>
<td>4.5</td>
</tr>
<tr>
<td>Sicily, Italy</td>
<td>36-39 N°</td>
<td>12-16 E°</td>
<td>2.5</td>
</tr>
<tr>
<td>South California</td>
<td>32-36.5 N°</td>
<td>114.75-121 W°</td>
<td>3.0</td>
</tr>
</tbody>
</table>
VII. CONCLUSION

In this research, the main approaches in application of machine learning methods to a problem of earthquake prediction are observed. The main open-source earthquake catalogs and databases are described. The definition of main metrics used for performance evaluation is given. A detailed review of published works is presented, which highlights the way of development of scientific methods in this area of research. Finally, during the discussion of the results achieved, further directions of research in the field of earthquake prediction are proposed. These are:

- Creating a “benchmark” earthquake dataset, which can be used to assess the quality of various predictor systems. The dataset includes frequently observed seismic zones and seismically active areas of East Asia and Europe, such as Central Japan and Sicily Island. The performance of previously proposed methods can also be evaluated using the «benchmark» dataset.

- Focusing on the most complex and important task of predicting earthquakes of high and extreme magnitudes (equal to or greater than 5.5).

- Making attempts to solve the problem of earthquake prediction in its original form, as determined by earthquake scientists; namely, the simultaneous specification of time, place and magnitude of seismic events with a certain probability.

REFERENCES


Construction of polarization kernels of size 16 for low complexity processing

Grigorii Trofimiuk, Peter Trifonov
ITMO University, Russia
Email: {gtrofimiuk,pvtrifonov}@corp.ifmo.ru

Abstract—An algorithm for construction of binary 16 × 16 polarization kernels with polarization rate 0.51828 which admit low complexity processing is proposed. The considered processing algorithm exploits linear relationship of the considered kernels and Arikan transform. The proposed approach relies on restricted application of elementary row operations to Arikan transform matrix, which are chosen to have minimal impact on complexity of the window processing algorithm.

The proposed construction resulted in relatively low number of kernels, which can be easily checked by computer-based search. Moreover, simulation results show that polar subcodes with obtained kernels can outperform polar codes with Arikan kernel, while having lower decoding complexity.

I. INTRODUCTION

Polar codes are a novel class of error-correcting codes, which achieve the symmetric capacity of a binary-input discrete memoryless channel W, have low complexity construction, encoding and decoding algorithms [1]. However, the performance of polar codes of practical length is quite poor. The reasons for this are the presence of imperfectly polarized subchannels and the suboptimality of the successive cancellation (SC) decoding algorithm. To improve performance, successive cancellation list decoding (SCL) algorithm [2], as well as various code constructions were proposed [3], [4], [5].

Polarization is a general phenomenon, and is not restricted to the case of Arikan matrix [6]. One can replace it by a larger matrix, called polarization kernel, which can provide higher polarization rate. Polar codes with large kernels were shown to provide asymptotically optimal scaling exponent [7]. Many kernels with various properties were proposed [6], [8], [9], [10]. Until recently, polar codes with large kernels were believed to be impractical due to very high decoding complexity.

The window processing algorithm for some 16 × 16 polarization kernels was introduced in [11]. This approach exploits the relationship between the considered kernels and the Arikan matrix. Essentially, the log-likelihood ratios (LLRs) for the input symbols of the considered kernels are obtained from the LLRs computed via the Arikan recursive expressions.

In this paper we present a construction method for polarization kernels, which admit efficient decoding by window based approach. The proposed method construct a class of polarization kernels, which are expected to be suitable for given processing method. The kernels are constructed by performing elementary row operations over Arikan transform matrix.

We show that with obtained kernels increasing list size in the SCL decoder provides much more significant performance gain compared to the case of Arikan kernel, and ultimately the proposed approach results in lower decoding complexity compared to the case of polar codes with Arikan kernel with the same performance.

II. BACKGROUND

A. Channel polarization

Consider a binary-input memoryless channel with transition probabilities \( W \{ y | c \}, c \in \mathbb{F}_2, y \in \mathcal{Y} \), where \( \mathcal{Y} \) is output alphabet. For a positive integer \( n \), denote by \([n]\) the set of \( n \) integers \( \{0, 1, \ldots, n-1\} \). A polarization kernel \( K \) is a binary invertible \( l \times l \) matrix, which is not upper-triangular under any column permutation. The Arikan kernel is given by

\[
F_m = \begin{pmatrix} 1 & 0 \\ 0 & I \end{pmatrix},
\]

where \( m \) is \( m \)-fold Kronecker product of matrix with itself.

An \((n = l^m, k)\) polar code is a linear block code generated by \( k \) rows of matrix \( G_m = M^{(m)} K \otimes I \), where \( M^{(m)} \) is a digit-reversal permutation matrix, corresponding to mapping \( \sum_{i=0}^{m-1} t_i l^i \rightarrow \sum_{i=0}^{m-1} t_{m-1-i} l^i, t_i \in [l] \). The encoding scheme is given by \( c_0^{n-1} = u_0^{n-1} G_m \), where \( u_i, i \in \mathcal{F} \) are set to some pre-defined values, e.g. zero (frozen symbols), \( |\mathcal{F}| = n - k \), and the remaining values \( u_i \) are set to the payload data.

It is possible to show that a binary input memoryless channel \( W \) together with matrix \( G_m \) gives rise to bit subchannels \( W_m (y_0^{n-1}, u_0^{i-1} | u_i) \) with capacities approaching 0 or 1, and fraction of noiseless subchannels approaching \( I(W) \) [6].

Selecting \( \mathcal{F} \) as the set of indices of low-capacity subchannels enables almost error-free communication. It is convenient to define probabilities

\[
W_m (u_i | y_0^{n-1}) = \frac{W_m (y_0^{n-1}, u_0^{i-1} | u_i)}{2W (y_0^{n-1})} = \sum_{u_0^{i-1}} \prod_{i=0}^{n-1} W ((u_0^{n-1} G_m)_i | y_i). \tag{1}
\]

Let us further define \( W_m (u_0^n) = W_m (u_0^n | y_0^n) \), where kernel \( K \) will be clear from the context. We also need...
probabilities \( W_{l}^{(j)}(y_0^n|y_0^{l-1}) = W_{l,F}^{(j)}(u_j^n|y_0^{l-1}) \) for Arikan matrix \( F_l \). Due to the recursive structure of \( G_m \), one has
\[
W_{m}^{(s+l)}(u_0^n|y_0^{n-1}) = 
\sum_{u_l^{(s+l-1)}} \prod_{j=0}^{l-1} W_{m-1}^{(s)}(\theta_K[u_0^{(s+l-1)}, j]|y_0^{j+l-1})
\]  
(2)
where \( \theta_K[u_0^{(s+l-1)}, j]_r = (u_r^{(r+1)}G_m)_j, r \in [s+1] \). A trellis-based algorithm for computing these values was presented in [12].

At the receiver side, one can successively estimate
\[
\hat{u}_i = \begin{cases} 
\arg \max_{u_i \in F} W_{m}^{(i)}(\tilde{y}_0^{i-1}, u_i|y_0^{i-1}), & i \notin F, \\
\text{the frozen value of } u_i, & i \in F.
\end{cases}
\]  
(3)
This is known as the successive cancellation (SC) decoding algorithm.

B. Rate of polarization
Let \( W : \{0, 1\} \rightarrow \mathcal{Y} \) be a symmetric binary-input discrete memoryless channel (B-DMC) with capacity \( I(W) \). By definition,
\[
I(W) = \sum_{y \in \mathcal{Y}} \sum_{x \in \{0, 1\}} \frac{1}{2} W(y|x) \log \frac{W(y|x)}{\frac{1}{2} W(y|0) + \frac{1}{2} W(y|1)}.
\]
Also, let \( Z(W) \subseteq \{0, 1\} \) denote the Bhattacharyya parameter of \( W \), i.e., \( Z(W) = \sum_{y \in \mathcal{Y}} \sqrt{W(y|0)W(y|1)} \).

Consider polarizing transform \( K^{\otimes m} \), where \( K \) is an \( l \times l \) polarization kernel, and bit subchannels \( W_{m,K}^{(i)}(y_0^{n-1}, u_0^{n-1}|u_i) \), induced by it. Let \( Z_{m,K}^{(i)} = Z(W_{m,K}^{(i)}(y_0^{n-1}, u_0^{n-1}|u_i)) \) be a Bhattacharyya parameter of \( i \)-th subchannel, where \( i \) is uniformly distributed on the set \([1^m] \). Then, for any B-DMC \( W \) with \( 0 < I(W) < 1 \), we will say that an \( \ell \times \ell \) matrix \( K \) has polarization rate \( E(K) \) if

(i) For any fixed \( \beta < E(K) \),
\[
\lim_{n \to \infty} \Pr[Z_n \leq 2^{-\beta n}] = I(W).
\]

(ii) For any fixed \( \beta > E(K) \),
\[
\lim_{n \to \infty} \Pr[Z_n \geq 2^{-\beta n}] = 1.
\]
That is, the rate of polarization shows how fast bit subchannels of \( K^{\otimes m} \) approach neither almost noiseless or noisy channel with \( n = m^3 \).

Suppose we constructed \( (n, k) \) polar code \( C \) with kernel \( K \). Let \( P_e(n) \) be a block error probability of \( C \) under transmission over \( W \) and decoding by SC algorithm. It was proven [6], that if \( n/k < I(W) \) and \( \beta < E(K) \), then
\[
P_e(n) \leq 2^{-\beta n}.
\]

It turns out that the rate of polarization is independent of channel \( W \). Namely, let \( \langle g_1, g_2, \ldots, g_k \rangle \) be a linear code, generated by vectors \( g_1, g_2, \ldots, g_k \). Let \( d_M(a, b) \) be the Hamming distance between \( a \) and \( b \). Let \( d_H(b, C) = \min_{c \in C} d_H(b, c) \) be a minimal distance between vector \( b \) and linear block code \( C \). We denote the \( i \)-th row of an \( \ell \times \ell \) matrix \( M \) as \( M[i], i \in [\ell] \).

The partial distances \( D_i, i = 0, \ldots, l-1, \ell \times l \) of the matrix \( K \) are defined as follows:
\[
D_i = d_H(K[i], (K[i+1], \ldots, K[l-1]), i = 0, \ldots, l-2,
D_{l-1} = d_H(K[l-1], 0).
\]
The vector \( D \) will be referred to as a partial distances profile. In work [6] it was shown that for any B-DMC \( W \) and any \( \ell \times \ell \) polarization kernels \( K \) with partial distances \( \{D_i\}_{i=0}^{l-1} \), the rate of polarization \( E(K) \) is given by
\[
E(K) = \frac{1}{\ell} \sum_{i=0}^{l-1} \log_2 D_i.
\]

The Arikan kernel \( F_1 \) has rate of polarization \( E(F_1) = 0.5 \), whereas random codes achieve \( E = 1 \). For polarization kernels of size 16 and 32 the kernels with rate of polarization 0.51828 and 0.53656 respectively can be obtained.

C. Scaling exponent
Let us fix a B-DMC \( W \) of capacity \( I(W) \) and a desired block error probability \( P_e \). Given \( W \) and \( P_e \), suppose we wish to communicate at rate \( I(W) − \Delta \) using a family of \( (n, k) \) polar codes with kernel \( K \). It has been shown that this value of \( n \) scales as \( O(\Delta^{-\mu(K)}) \), where the constant \( \mu(K) \) is known as the scaling exponent [8].

The scaling exponent depends on channel. Unfortunately, the algorithm of its computing is only known for the case on binary erasure channel (BEC) [13],[8].

The Arikan kernel \( F_1 \) has \( \mu(K) = 3.627 \), whereas random codes achieve optimal \( \mu = 2 \). The best known scaling exponent for \( 16 \times 16 \) polarization kernel is 3.346 [11].

D. Computing kernel input symbols LLRs

1) General case: Our goal is to compute probabilities \( W_{1}^{(j)}(u_0^n|y_0^{n-1}) \) for a given polarization transform \( K^{\otimes m} \). Let us assume for the sake of simplicity that \( m = 1 \). The corresponding task will be referred to as kernel processing.

We propose to introduce approximate probabilities
\[
W_{1}^{(j)}(u_0^n|y_0^{n-1}) = \max_{u_j^{l-1}} W_{1}^{(l-1)}(u_0^{l-1}|y_0^{l-1})
\]
\[
= \max_{u_j^{l-1}} \prod_{i=0}^{l-1} W((u_0^{l-1}K)_i|y_i).
\]

This is the probability of the most likely continuation of path \( u_0^n \) in the code tree, without taking into account possible freezing constraints on symbols \( u_i, i > j \). Note that the same probabilities were introduced in [14], [15], and shown to provide substantial reduction of the complexity of sequential decoding of polar codes.
Decoding can be implemented using the log-likelihood ratios $S_{m,i} = S^{(i)}_m(u^{l-1}_0|y^{n-1}_0) = \ln \frac{W^{(i)}_m(v^{l-1}_1|y^{n-1}_0)}{W^{(i)}_m(v^{l-1}_0|y^{n-1}_0)}$. Hence, kernel output LLRs $S_{1,i}, i \in [l]$ can be approximated by

$$S_{1,i} \approx S_{1,i} = \ln \frac{\hat{W}^{(i)}_1(u^{l-1}_0|0|y^{l-1}_0)}{\hat{W}^{(i)}_1(u^{l-1}_0|1|y^{l-1}_0)} = \max_{u^{l-1}_i} \ln W^{(l-1)}_1(u(0)|y^{l-1}_0) - \max_{u^{l-1}_i} \ln W^{(l-1)}_1(u(1)|y^{l-1}_0),$$

(6)

where $u(a)^i = (u^{l-1}_0.a.n^{l-1}_0)$. The above expression means that $S_{1,i}$ can be computed by performing ML decoding of the code, generated by last $l-i+1$ rows of the kernel $K$, assuming that all $u_j, i < j < l$, are equiprobable.

2) Window processing: Straightforward evaluation of (6) for arbitrary kernel has complexity $O(2^l)$. However, we have a simple explicit recursive procedure for computing these values for the case of the Arikan transform $F_i$.

Let $l = 2^i$. Consider encoding scheme

$$v^{l-1}_0 = v^{l-1}_i F_i.$$

(7)

Similarly to (5), define approximate probabilities

$$\hat{W}^{(i)}_t(v^{l-1}_0|y^{l-1}_0) = \max_{v^{l-1}_i} W^{(l-1)}_t(v^{l-1}_0|y^{l-1}_0)$$

and modified log-likelihood ratios

$$S^{(i)}_t(v^{l-1}_0, y^{l-1}_0) = \log \frac{\hat{W}^{(i)}_t(v^{l-1}_0|0|y^{l-1}_0)}{\hat{W}^{(i)}_t(v^{l-1}_0|1|y^{l-1}_0)}.$$

It can be seen that

$$S^{(2i)}_1(v^{2i-1}_0, y^{2i-1}_0) = \sigma(a) \sigma(b) \min(|a|, |b|),$$

(8)

$$S^{(2i+1)}_1(v^{2i+1}_0, y^{2i+1}_0) = (-1)^{v_1} a - b,$$

(9)

where $N = 2^k$, $a = S^{(i)}_1(v^{2i-1}_0, y^{2i-1}_0, v^{2i-1}_0, y^{2i-1}_0)$, $b = S^{(i)}_1(v^{2i-1}_0, y^{2i-1}_0)$. Then the log-likelihood of a path $v^{l-1}_0$ can be obtained as [16]

$$R(v^{l-1}_0|y^{l-1}_0) = \log \hat{W}^{(l-1)}_1(v^{l-1}_0|y^{l-1}_0) = R(v^{l-1}_i|v^{l-1}_i) + \tau \left( S^{(i)}_t(v^{l-1}_0, y^{l-1}_0), v^{l-1}_i \right),$$

(10)

where $R(c|y^{l-1}_0)$ can be set to 0, $c$ is an empty sequence, and

$$\tau(S, v) = \begin{cases} 0, & \sigma(S) = (-1)^v \\ -|S|, & \text{otherwise.} \end{cases}$$

It was suggested in [17] and [11] to express values $W^{(i)}_1(v^{l-1}_h|y^{l-1}_0)$ via $W^{(i)}_1(v^{l-1}_0|y^{l-1}_0)$ for some $j$. Indeed, $T = F_i$, where $T$ is an $l \times l$ matrix. Let

$$v^{l-1}_0 = u^{l-1}_0 F_i = u^{l-1}_0 K \Rightarrow u^{l-1}_0 = v^{l-1}_0 T.$$

Observe, that it is possible to reconstruct $u^{l-1}_i$ from $v^{l-1}_i$, where $\tau_i$ is the position of the last non-zero symbol in the $i$-th column of $T$. For the sake of simplicity we assume that all $\tau_i, i \in [l]$ are distinct. The general case is considered in [11].

Indeed, vectors $u^{l-1}_0$ and $v^{l-1}_0$ satisfy the equation

$$u^{l-1}_i = \sum_{j=0}^{l-1} v^{l-1}_j T[j, i],$$

(11)

where $T[j, i]$ is a $j$-th element of row $T[i]$.

Let $h_i = \max_{j \in [i+1]} \tau_j$. It can be seen that

$$W^{(j)}_1(v^{l-1}_0|y^{l-1}_0) = \sum_{v^{h_j}_0 \in Z_j} W^{(h_j)}_1(v^{h_j}_0|y^{l-1}_0),$$

(12)

where $Z_j$ is the set of vectors $v^{h_j}_0$, such that (11) holds for $i \in [j]$. Similarly we can rewrite the above expression for the case of the approximate probabilities

$$\hat{W}^{(j)}_1(v^{l-1}_0|y^{l-1}_0) = \max_{v^{h_j}_0 \in Z_j} \hat{W}^{(h_j)}_1(v^{h_j}_0|y^{l-1}_0),$$

(13)

where $v^{h_j}_0 = \hat{W}^{(h_j)}_1(v^{h_j}_0|y^{l-1}_0)$.

Let $Z_{i, b} = \{ v^{h}_0 | v^{h}_0 \in Z_i \}$, where $u_i = b$. Hence, one obtains

$$S_{1,i} = \max_{v^{h}_0 \in Z_{i, 0}} R(v^{h}_0|y^{l-1}_0) - \max_{v^{h}_0 \in Z_{i, 1}} R(v^{h}_0|y^{l-1}_0).$$

(14)

Observe that computing these values requires considering multiple vectors $v^{h}_0$ of input symbols of the Arikan transform $F_i$. Let

$$D_i = [h_i + 1]\{ \tau_0, \tau_1, \ldots, \tau_i \}$$

be a decoding window, i.e. the set of indices of independent (from $u^{l-1}_0$) components of $v^{h}_0$. Note that $|D_i| = |[h_i + 1]| \{ \{ \tau_0, \tau_1, \ldots, \tau_i \} = h_i + 1 - (i + 1) = h_i - i$ since all $\tau_i$ are distinct and $\{ \tau_0, \tau_1, \ldots, \tau_i \} \subseteq [h_i + 1]$. The calculation of LLRs $S_{1,i}$ via (14) will be referred to as the window processing algorithm.

The number of path scores to be computed in (14), which determines the processing complexity, is equal to $2|D_i| + 1$. Let $M(K)$ denotes the $\max_{e \in [l]} |D_i|$. In general, one has $M(K) = O(l)$ for an arbitrary kernel $K$.

3) Complexity: The complexity of LLR $S_{1,i}$ computation via the straightforward implementation of the window processing algorithm (14) consist of the several components. In this work we count the arithmetical complexity as a number summation and comparison operations, which is considered to be equal.

At first, to compute $S_{1,i}$, one should obtain path scores $R(v^{h}_0|y^{l-1}_0), v^{h}_0 \in Z_i$. According to the expression (10), the path score $R(v^{h}_0|y^{l-1}_0)$ is equal to

$$R(v^{h}_0|y^{l-1}_0) = \tau \left( S^{(h_i)}_t(v^{h}_0, y^{l-1}_0), v^{h}_0 \right).$$

If one stores the intermediate results of (8) and (9), then the complexity of computing $S^{(h_i)}_t = S^{(h_i)}_t(v^{h}_0, y^{l-1}_0)$ is

1The method given in [10] is a special case of this approach.
given by $2^B(h_i) - 1$ operations, where $B(h)$ is a position of the last nonzero digit in the binary representation of $h$, i.e. $h = 2^{b_0} + 2^{b_1} + \cdots + 2^{B(h)}$. If $h = 0$ then $B(h)$ is assumed to be $t$.

Totally, $2^{D_t}$ LLRs $S_t^{(h_i)}$ should be computed. Then, for LLR $S_t^{(h_i)}$ and $v_i \in [1]$ one should calculate the value $\tau \left( S_t^{(h_i)}(v_i^{-1}, y_0^{-1}), v_i \right)$, which can be done in one summation. In sum, it gives $2^{D_t}$ operations more. Moreover, if $h_i - h_{i-1} > 1$, then the above described computations should be done for LLRs $S_t^{(h_i)}$, $h_{i-1} < h \leq h_i$. It can be observed, that the number of such LLRs is given by $2^{D_t} - (h_i - h) = 2^{h_i} - 1$.

In total, the complexity of path scores $R(v_0^{h_i}|y_0^{l-1}), v_0^{h_i} \in \mathcal{Z}_i$, calculation is given by

$$\Lambda(i) = \sum_{h_{i-1}+1}^{h_i} \left( 2^{h_i - i} (2^B(h) - 1) + 2^{h_i - i} \right) = \sum_{h_{i-1}+1}^{h_i} 2^{h_i + B(h) - i}$$

and $h_{i-1}$ is assumed to be $-1$.

To complete the LLR $S_{14}$ computation, the corresponding maximum of path scores should be computed, which requires $2^{D_{14} + 1}$ comparisons.

Note that in the case of $h_i = h_{i-1}$ we assume that all path scores are stored together with corresponding partial maximums, thus, one substruction needed only.

In sum, the complexity of the straightforward implementation of the window processing algorithm for kernel $K$ can be estimated as

$$\Psi(K) = \sum_{i=0}^{l-1} \Phi(i),$$

(16)

where

$$\Phi(i) = \begin{cases} \left( 2^{h_i - i + 1} + \Lambda(i), \quad h_i > h_{i-1}, \right. \\ 1, \quad \text{otherwise.} \end{cases}$$

III. CONSTRUCTION OF POLARIZATION KERNELS

Our goal is to construct polarization kernels with polarization rate greater that 0.5, which admit low complexity processing. Such rate of polarization rate can be achieved for kernels of size $l = 16$ and $l \geq 23$ [6]. In this work we focus on $16 \times 16$ polarization kernels.

The maximum rate of polarization among $16 \times 16$ kernels is equal to 0.51828, which can be achieved by the kernel with the partial distances profile

$$\mathbb{D}^{(*)} = [1, 2, 2, 2, 2, 4, 4, 4, 4, 6, 6, 8, 8, 8, 8, 16].$$

There are polarization kernels with partial distance profile which corresponds to some permutation of $\mathbb{D}^{(*)}$, what will be demonstrated later, but the complete list of such permutations is unknown. Therefore, it is convenient to begin our investigation with kernels with monotonic increasing partial distances.

The minimization of the complexity (16) by the exhaustive search among all polarization kernels $K$ of size $16 \times 16$ and partial distances $\mathbb{D}^{(*)}$ is intractable. Therefore, we are going to significantly reduce the search space to some restricted class of polarization kernels, which are expected to have moderate $\Psi(K)$.

A. Row permutation

Recall that $\tau_i$ is the position of the last non-zero symbol in the $i$-th column of $T = F_4 K^{-1}$, $h_i = \max \{ \tau_i \}$ and $|D_{16}| = h_i - i$.

It can be seen, that the value of $h_i - i$ increases once $\tau_i > i$ appears in $T$, therefore the heuristic minimization of $\Phi(K)$ can be done with minimization of $|\tau_i - i|$, $i \in [l]$.

The minimal value of $|\tau_i - i| = 0$ is achieved by Arikan transform $F_4$. The kernel with partial distances $\mathbb{D}^{(4)}$ can be derived by performing elementary operations over rows space of $F_4$, since $F_4$ is invertible.

The partial distance profile of the Arikan transform $F_4$ is given by

$$\mathbb{D}^{(F_4)} = [1, 2, 2, 4, 4, 4, 4, 4, 4, 8, 8, 8, 8, 16].$$

Hence, we can begin construction procedure with row permutation of the matrix $F_4$.

Let $P_\rho$ be a permutation matrix, which corresponds to the permutation

$$\rho = \left( \begin{array}{cccc} 0 & 1 & \cdots & 14 \\ \rho(0) & \rho(1) & \cdots & \rho(14) \end{array} \right).$$

For convenience, we enumerate elements of $\rho$ from zero unlike standard notation. For brevity we will write $[\rho(1), \rho(2), \ldots, \rho(16)]$. Consider the kernel $K_\rho = P_\rho F_4$, consequently,

$$T = F_4 K_\rho^{-1} = F_4 (P_\rho F_4)^{-1} = P_\rho^T.$$

Thus, $\tau_i, i \in [l]$ are given by $\rho(i)$. Therefore, the processing complexity for $K_\rho$ directly depends on the permutation $\rho$.

We start our construction with permuted Arikan kernels $K_\beta$ given by permutations

$$\beta = [0, 1, 2, 4, 8, w_0, w_1, w_2, w_3, w_4, w_5, 7, 11, 13, 14, 15],$$

where $w$ is an arbitrary permutation of the vector $[3, 5, 6, 9, 10, 12]$. The indices of $w$ are the indices of $F_4$ rows with Hamming weight 4. The obtained kernels have monotonic partial distance profile

$$\mathbb{D}^{(4)} = [1, 2, 2, 2, 2, 4, 4, 4, 4, 4, 8, 8, 8, 8, 16].$$

For instance, the permutation

$$\sigma = [0, 1, 2, 4, 8, 3, 5, 6, 9, 10, 12, 7, 11, 13, 14, 15]$$

results in the permuted Arikan kernel $K_\sigma$ with $E(K_\sigma) = 0.5$ and scaling exponent $\mu(K_\sigma) = 3.479$ [10]. It can be observed, that kernel $K_\sigma$ has the least processing complexity $\psi(K)$ among all permuted $F_4$ kernels which have the partial distance profile $\mathbb{D}^{(4)}$. The maximal $h_i$ for this kernel is given by 4, which results in relatively low complexity.
B. Row addition

To transform the kernel $K_β$ into the kernel with partial distance $D(∗)$, one should sum rows of $K_β$. It is proven [8], that addition of row $K_i$ to row $K_j$ with $i > j$ does not change the properties of the kernel $K$. Thus, we consider row additions with $i < j$ only.

The addition of two rows can also increase the maximal size of the decoding windows. Indeed, let $X_{i,j}$ be an elementary matrix which corresponds to addition of row $i$ to row $j$. In other words, $X_{i,j}$ is an identity matrix with $X_{i,j}[j,i] = 1$.

Then

$$K = X_{i,j}P_ρF_4 \Rightarrow T = P^T_ρX_{i,j},$$

which means that the column $j$ has been added to the column $i$ of the matrix $P^T_ρX_{i,j}$. After row addition in $K$, $τ_i = \max(τ_i, τ_j)$, which can increase the $τ_i - i$. It can lead to increasing of the size of the corresponding decoding window. It means that one should use addition matrices $X_{i,j}$ with as small as possible values $|j - i|$.

To keep the processing complexity as small as possible, we suggest to sum only rows $K_β[i]$, $i \in \{5, 6, 7, 8, 9, 10\}$ to each other. These rows have a Hamming weight 4. It was shown that sum $x$ of these rows can produce vectors of weight $≥ 6$ and, furthermore, there exist several such $x$ as

$$d_H(x, \langle K_β[11], K_β[12], \ldots, K_β[15]\rangle) = 6$$

(see [18] page 429).

C. The construction algorithm

Let $M = \{3, 5, 6, 9, 10, 12\}$. We propose to minimize the decoding window processing complexity over set $K$ of $16 \times 16$ kernels, which is given by following constraints on kernel $K$:

- $K[i] = K_β[i]$, $i \in \{0, 1, 2, 3, 4, 11, 12, 13, 14, 15\}$,
- $K[9], K[10] \in V_0$, where $V_0 = \{c \in C | d_H(c, 0) = 6\}$, $0$ is a zero element vector and $C = \{\langle F_i[i], i \in M\rangle\}$,
- $K_j \in V_1$, where $V_1 = \{F_i[i], i \in M\}, j \in \{5, 6, 7, 8\}$.

The above construction results in the search space of size

$$|K| = |V_0|^2 \cdot |V_1|^4 = 27^2 \cdot 6^4 = 944784.$$ 

It is easy to observe, that the proposed construction can produce kernels with partial distances distinct from $D(∗)$ and even to singular matrices. However, one does not need to compute the complete partial distance profile $D$ for $K \in K$, because the kernel $K$ can be dropped once its partial distance does not match the $D(∗)$. Of course, there are a lot of possible methods for reduction of $K$, however, there is no need for them since computer-based search over $K$ runs in several minutes.

IV. NUMERIC RESULTS

A. Kernel construction

1) Monotonic partial distances: Computer-based search results in set $K_β$ of 60480 $16 \times 16$ polarization kernels $K$ with $E(K) = 0.51828$. For each $K$ in $K_β$, we compute its complexity $Ψ(K)$. Moreover, we also computed the BEC scaling exponent $μ(K)$ for each kernel. The scaling exponent also affects on the error correction performance, so we write the minimal processing complexity for kernel with different scaling exponent.

Table I demonstrates all occurred scaling exponents of kernels from the set $K_β$ together with minimal processing complexity. Furthermore, for each presented scaling exponent the kernel $K$ with $M(K) = 4$ is provided. It can be seen that the minimal complexity of 660 operations is provided by kernels with $μ(K) = 3.363$ and kernels with the lowest $μ(K) = 3.346$ requires the maximal complexity among other scaling exponents.

It turns out, that the complexity of window processing can be significantly reduced. For instance, the kernel $K_1$, illustrated in Figure 1, $K_1 \in K_β$, $μ(K_1) = 3.346$, was reported in [11] to have processing complexity of 472 arithmetic operations instead of 740.

For comparison, the general trellis-based algorithm [12] applied to processing of $K_1$ kernel has the complexity of 7530 operations, which is 10 times higher compared to the complexity of straightforward window processing algorithm. However, minimization of maximal size of the decoding windows is crucial, as far as complexity grows exponentially with it. For instance, $16 \times 16$ BCH kernel

$$K_{BCH} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix},$$

with $E(K_{BCH}) = 0.51828$ and $μ(K_{BCH}) = 3.396$, which consist of the sequence of nested generator matrices of extended BCH codes, has $M(K_{BCH}) = 12$ and the
processing complexity \( \Psi(K_{BCH}) = 72563 \). Whereas the algorithm [12] for \( K_{BCH} \) requires 12456 operations. This example shows us the importance of minimization of decoding windows sizes.

2) Permuted partial distances: In the previous section we showed how to find kernels of size 16 with monotonic partial distance profile \( \mathbb{D}^{(\ast)} \). It resulted in kernels with the \( \mathcal{M}(K) = 4 \). For further complexity reduction we are going to perform row permutations over row space of the obtained kernels, which preserves the polarization rate.

Given kernel \( K \), the value \( \mathcal{M}(K) \) can be reduced by row permutation of \( K \). By step-by-step exchange of the kernel rows, we performed an heuristic search of row permutation, which preserve the polarization rate of \( K_1 \).

Table II demonstrates the properties of kernels which we obtained by permutations of the kernel \( K_1 \). It can be observed, that higher scaling exponent requires lower processing complexity, furthermore, the maximal size of the decoding windows can be also reduced for kernels with polarization rate 0.51828. For instance, the kernel \( K_2 \), illustrated in Figure 1, has \( E(K_2) = 0.51828, \mu(K) = 3.45 \) and \( \mathcal{M}(K) = 3 \). The kernel \( K_2 \) is given by \( \tilde{\rho}K_1 \), where

\[
\tilde{\rho} = [0, 1, 2, 7, 3, 4, 5, 6, 9, 10, 11, 12, 8, 13, 14, 15],
\]

and has a partial distance profile

\[
\tilde{\mathbb{D}} = [1, 2, 4, 2, 2, 4, 6, 8, 8, 4, 8, 16],
\]

which is not monotonic unlike \( \mathbb{D}^{(\ast)} \). It was shown in [11] that the kernel \( K_2 \) can be processed with 183 operations instead of 293 operations in straightforward implementation.

Unfortunately, we do not have a proof that the kernel \( K_2 \) admits minimum possible complexity of window processing algorithm among all \( 16 \times 16 \) polarization kernels with polarization rate 0.51828.

### B. Performance of polar codes with the constructed kernels

We constructed \((4096, 2048)\) polar codes with kernels \( K_1 \) and \( K_2 \), obtained by the proposed construction, and investigated their performance for the case of AWGN channel with BPSK modulation. The sets of frozen symbols were obtained by Monte-Karlo simulations.

Figure 2 illustrates the performance of plain polar codes and polar subcodes [4],[19]. It can be seen that the codes based on kernels \( K_1 \) and \( K_2 \) with improved polarization rate \( E(K_1) = E(K_2) = 0.51828 \) provide significant performance gain compared to polar codes with Arikan kernel. Moreover, polar subcodes with kernels \( K_1, K_2 \) under SCL with \( L = 8 \) have almost the same performance as polar subcodes with Arikan kernel under SCL with \( L = 32 \). Observe also that the codes based on kernels with lower scaling exponent exhibit better performance despite of the fact that scaling exponent is computed for the BEC.

Figure 3 presents simulation results for \((4096, 2048)\) polar subcodes with different kernels under SCL with different \( L \) at \( E_b/N_0 = 1.25 \) dB. It can be seen that the kernels with polarization rate 0.51828 require significantly lower list size \( L \) to achieve the same performance as the code with the Arikan kernel. Moreover, this gap grows with \( L \). This is due to improved rate of polarization, which results in smaller number of unfrozen imperfectly polarized bit subchannels. The size of the list needed to correct possible errors in these subchannels grows exponentially with their number (at least for the genie-aided decoder considered in [20]). On the other hand, lower
scaling exponent gives better performance with the same list \( L \), but the slope of the curve remains the same for both kernels \( K_1, K_2 \).

Figure 4 presents the same results in terms of the actual decoding complexity. Recall that proposed kernel processing algorithm uses only summations and comparisons. The SCL algorithm was implemented using the randomized order statistic algorithm for selection of the paths to be killed at each phase, which has complexity \( O(L) \). Observe that the polar subcode based on kernel \( K_2 \) can provide better performance with the same decoding complexity for \( \text{FER} \leq 8 \cdot 10^{-3} \). This is due to higher slope of the corresponding curve in Figure 3, which eventually enables one to compensate relatively high complexity of the LLR computation algorithm presented in \[11\].

Unfortunately, \( K_1 \) kernel, which provides lower scaling exponent, has greater processing complexity than \( K_2 \), so that its curve intersects the one for the Arikan kernel only at \( \text{FER} = 2 \cdot 10^{-3} \).

V. CONCLUSIONS

In this paper the construction method for \( 16 \times 16 \) polarization kernels with polarization rate 0.51828 were proposed. These kernels admits low complexity decoding by window processing algorithm. The construction method performs elementary operations over row space of the Arikan transform matrix. These elementary operations are chosen to have minimal impact on the complexity of the window processing algorithm.

It was shown that in the case of SCL decoding with sufficiently large list size, the constructed kernels results in lower decoding complexity compared to the case of polar (sub)codes with Arikan kernel with the same performance.

Extension of the proposed construction to the case of kernels with larger size remains an open problem.

REFERENCES

A Quantitative Study of Two Matrix Clustering Algorithms

Alexander Slesarev
Saint-Petersburg University, Russia
alexander.g.slesarev@gmail.com

Viacheslav Galaktionov¹,²
JetBrains Research,
1 Saint-Petersburg State University, Russia
viacheslav.galaktionov@gmail.com

Nikita Bobrov¹,²
JetBrains Research,
1 Saint-Petersburg State University, Russia
nikita.v.bobrov@gmail.com

George Chernishev¹,²
JetBrains Research,
1 Saint-Petersburg State University, Russia
g.chernyshev@spbu.ru

Abstract—Matrix clustering is a technique which permutes rows and columns of a matrix to form densely packed regions. It originated in the 70’s and initially was used for various object grouping problems, such as machine-component grouping. The database community noticed these algorithms and successfully applied them to the vertical partitioning problem. Recently, there has been a resurgence of interest in these algorithms. Nowadays, they are being considered for dynamic (on-line) vertical partitioning and tuning of multistores.

In our previous papers we have described our project aimed at studying the applicability of recent matrix clustering algorithms for the vertical partitioning problem. We have presented our evaluation approach and reported results concerning several of these algorithms. Our idea was to evaluate them directly using the PostgreSQL database. Previous studies have found that these algorithms can be of use if they employ the attribute replication strategy. In this paper, we continue our investigation and consider a novel algorithm of this class. Its distinctive feature is that it performs attribute replication during the branch and bound search. We compare it with the best one of the earlier algorithms using both real and synthetic workloads.

Our experiments have demonstrated that the novel algorithm produces slightly worse configurations (about 10%), but its run times are significantly better and are almost independent of the cohesion parameter.

Index Terms—databases, database tuning, physical design, vertical partitioning, experimentation, matrix clustering, fragmentation.

I. INTRODUCTION

Vertical partitioning is a technique used to speed up query processing in databases. Its core idea is dividing a table into fragments which contain only a subset of attributes. In order to ensure that the database will not undergo semantic changes, the following rules of vertical partitioning are used [1]: completeness, reconstruction, and disjointness. Sometimes the disjointness rule is relaxed. In this case, it is said that vertical partitioning is performed with attribute replication.

The speedup comes from the fact that some queries would have to read less data. Indeed, suppose that for a given query all needed attributes are allocated into a single fragment, and this fragment contains no extra attributes. In this case, one can roughly estimate that number_of_rows × extra_attributes_lengths bytes can be saved during the data reading phase in case of a slotted page data layout [2].

However, if there is a query that requires attributes from two or more fragments, then its performance may suffer due to the record reconstruction costs. Data modification operations (inserts, deletes, and updates) complicate things further since they involve all attributes of a record and thus, all fragments should be modified. The impact of additional disk seeks on a hard drive may be so large that it can make the partitioning scheme impractical.

Due to all these facts, there is still no support of fully-automatic vertical partitioning in industrial database systems. Moreover, unlike the horizontal, vertical partitioning is not supported in SQL DDL: e.g., in PostgreSQL it is possible to define horizontal fragments using the “PARTITION BY” clause for a “CREATE TABLE” statement.

Nevertheless, there are multiple semi-automatic stand-alone tools (“advisors”, see surveys [3], [4]) for this task. All of them recommend beneficial vertical partitioning schemes for a specified workload (queries) and let the database administrator decide whether to implement them or not.

The reason for the limited success of these tools (the overwhelming majority of them are academic research prototypes and not industrial products) is that finding an optimal solution is an NP-hard problem for many different formulations [5]–[7]. Another well-known fact is that the number of different vertical partitioning schemes for a single table is equal to the Nth Bell number, where N is the number of attributes [8]. Nevertheless, due to the interest of both industrial and academic communities, the development of such advisors continues.

In the core of such a system lies an algorithm that traverses the partitioning space and selects a beneficial scheme. There
are two classes of algorithms for this task: cost-based and heuristic. The former employ some kind of a cost-based model to evaluate the quality of a given partitioning scheme in terms of query run times, required space, and other metrics. The latter proposes some kind of procedure to generate a “good” scheme. Usually, some considerations are presented as to why it is likely to generate a beneficial partitioning scheme, but not a strict proof.

The heuristic approach was very popular in the 70’s and 80’s, but later was abandoned in favour of the cost-based one. Nowadays, there is a resurgence of interest in heuristic approaches due to the appearance of novel application areas: dynamization of vertical partitioning [9]–[12], tuning of multistores [13], big data applications or any other cases featuring limited resources.

In our previous studies [14]–[16] we have described our project that aims to study the applicability of several recently developed matrix clustering algorithms. Our project is motivated by the fact that the authors of these algorithms have not evaluated their performance (run times, quality) using a DBMS and a workload. To address this, we have constructed a framework for evaluating such algorithms that uses PostgreSQL. Then we have evaluated a number of these algorithms [17]–[19] using the TPC-H benchmark. In this paper, we continue our research and consider the most recent algorithms of this type [20].

The rest of this paper is organized as follows. In Section II we provide a short introduction into the subject and describe existing types of heuristic approaches. Next, in Section III we introduce matrix clustering algorithms and provide a description of the considered algorithm. Section IV describes our experimental framework, setup, and the experiments. The results of evaluation are discussed in Section V, threats to validity of this study are presented in Section VI and Section VIII concludes this paper.

II. RELATED WORK

As it was stated in the Introduction, there are two types of approaches to the vertical partitioning problem — cost-based and heuristic. Since this problem is almost 40 years old, and a lot of results have been accumulated, we will only describe studies on heuristic algorithms in this section. More extensive surveys that examine cost-based approaches as well can be found in references [4], [21]. Heuristic vertical partitioning algorithms can be classified into the following major groups [14], [15]:

- Attribute affinity and matrix clustering approaches [17]–[19], [22]–[24]. Attribute affinity is a measure which shows how frequently two attributes are requested together in a given workload. These approaches use it as follows:

  1) A workload is used to construct an Attribute Usage Matrix (AUM), a special way to represent which attributes are used by each query of a workload.

  2) Attribute affinity is calculated for all pairs of attributes and an Attribute Affinity Matrix (AAM) is constructed.

  3) A special algorithm for row and column permutation is applied to the AAM. Afterwards, “dense” regions are extracted and used to define resulting partitions.

Studies employing the matrix clustering approach (and in particular, the ones considered in our paper) permute AUMs, but not AAMs.

- Graph approaches [5], [25]–[28]. Similarly to the previous type, these approaches start with a workload and use it to construct an AAM. However, in this case the AAM is considered as an adjacency matrix of an undirected weighted graph, where the nodes are attributes and the edge weights show the affinity for a given pair of attributes. Finally, this graph is used to search for special structures which will be used to define resulting partitions. There are many approaches, e.g. Kruskal-like algorithms or cutting the Hamiltonian way.

- Data mining approaches [29]–[31]. In this type of approach, association rule mining is used to derive vertical fragments. The workload is considered as a transaction set, and the rules use sets of attributes as items. This group of vertical partitioning algorithms is relatively new, so existing algorithms for association rule search are frequently used. For example, a popular choice is to adapt Apriori [32] or FP-Max algorithms.

III. MATRIX CLUSTERING ALGORITHMS

A. Basics

The general scheme of this approach is as follows [14], [15]:

- Construct an Attribute Usage Matrix (AUM) from the workload. The matrix is defined as follows:

\[
M_{ij} = \begin{cases} 
1, & \text{query } i \text{ uses attribute } j \\
0, & \text{otherwise}
\end{cases}
\]

- Cluster the AUM by permuting its rows and columns to obtain a block diagonal matrix.

- Extract these blocks and use them to define the resulting partitions.

Some approaches do not operate on a 0-1 matrix. Instead, they modify matrix values to account for additional information like query frequency, attribute size and so on. Let us consider an example. Suppose that there are six queries accessing six attributes:

q1: SELECT a FROM T WHERE a > 10;
q2: SELECT b, f FROM T;
q3: SELECT a, c FROM T WHERE a = c;
q4: SELECT a FROM T WHERE a < 10;
q5: SELECT e FROM T;
q6: SELECT d, e FROM T WHERE d + e > 0;

The next step is the creation of an AUM using this workload. The resulting matrix is shown in Figure 1a. After the application of a matrix clustering algorithm, the reordered
Fig. 1: Matrix clustering algorithm

(a) AUM

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>q2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>q3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>q4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>q5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>q6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(b) Reordered AUM

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>c</th>
<th>b</th>
<th>f</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>q1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>q3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>q4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>q2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>q5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 2: Non-decomposable matrix

AUM (Figure 1b) is acquired. The resulting fragments are the following: (a, b), (b, f), (d, e).

However, not all matrices are fully decomposable. Consider the matrix presented in Figure 2. The first column obstructs the perfect decomposition into several clusters. In this case, the algorithm should produce a decomposition which minimally harms query processing and results in an overall performance improvement. Matrix clustering algorithms employ different strategies to select such a decomposition.

A systematic review of matrix clustering algorithms is presented in studies [14], [15]. Here, we will consider only the recent approaches.

B. Recent Advances

Within our project, we study a series of works by Chun-Hung Cheng et al. [17]–[20]. These algorithms employ a branch and bound search that tries to find submatrices that conform to specific conditions. Their input is the threshold (target cohesion), which is defined as the share of 1’s in the resulting matrices.

In this study we are interested in two algorithms — A09 [19] and A11 [20].

The A09 algorithm comes with three different strategies that define the treatment of intersubmatrix attributes (the ones that were marked as obstacles to decomposition) — nearest, separate, and replicate. In the first one such attribute goes to the nearest submatrix, in the second all such attributes are assigned to a dedicated submatrix, and the last one replicates the attribute into each submatrix that requires it. Note that the strategy is applied after the clustering is done.

The A11 algorithm has a different idea. If during branch and bound traversal the algorithm encounters such an attribute, then it replicates it and tries to decompose the matrices further.

IV. Experiments

A. Benchmarking

In our previous works we have developed a special prototype for experimental evaluation of matrix clustering algorithms. The idea of our approach is to directly check whether the generated partitioning schemes help to improve query performance. For these purposes we employ the PostgreSQL DBMS and several workloads, both real and synthetic.

The architecture of our prototype is presented in Figure 3. It consists of the following modules:

- The parser reads the workload from a file. It extracts the queries and passes them to the executor, so that their execution times can be measured. It also constructs the AUM, which serves as input for the selected algorithm.
- The algorithm identifies clusters and passes that information to the partitioner to create corresponding temporary tables.
- The query rewriter also receives this information. It replaces the name of the original table with the ones that were generated by the partitioner.
- The partitioner generates new names and sends partitioning commands to the database. The exact commands are SELECT INTO and ALTER TABLE. The latter lets it transfer primary keys.
- The executor accepts queries and sends them to PostgreSQL to measure the time of execution.

B. Experimental Setup and Evaluation Procedure

In our experiments, we have used the following hardware and software setup:

- Inspiron 15 7000 Gaming (0798), 8GiB, Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz, TOSHIBA 1TB MQ02ABD1
- Ubuntu 18.10, PostgreSQL 11.1, gcc 8.2.0

Data for quality-related graphs was obtained by running 10 invocations of the respective algorithm and averaging the result. We deemed a single run sufficient for run time graphs, since even one invocation can require up to two hours.

In order to ensure maximum quality of experiments, several measures were taken:

1) We eliminated data caching for both operating system caches and PostgreSQL caches. For this, we
have restarted PostgreSQL and dropped the operating system caches before running each query. Operating system caches were dropped by writing “3” to /proc/sys/vm/drop_caches.

2) Next, we manually checked plans for each query and noticed that some queries may have different scan operator implementations depending on the table. Frequently, a query on a partitioned table did not have a sequential scan, but rather parallel. To handle this, we have restricted the query optimizer to use only sequential scans by issuing the following command set max_parallel_workers_per_gather to 0;

To ensure that no hidden caching or other unaccounted processes happen, we have designed the following simple criterion. Suppose that we have a set of queries that involve only a single table and are essentially scans without complex data processing. Initially, we run these queries on the original table and record their run times. Then, for every query we designate a table that will contain all attributes necessary to evaluate it. Thus, no joins are needed. At the same time, for some queries, the tables assigned to them will also contain extra attributes. Therefore, some tables may serve more than one query. Then we run each query on corresponding table and record its run time. Eventually the following two values should be approximately equal:

1) \[ \sum_{q_i \in \text{Queries}} \frac{\text{size}(T)}{\text{time}(q_i)} \]

2) \[ \sum_{q_i \in \text{Queries}} \frac{\text{size}(\text{table}(q_i))}{\text{time}(q_i)} \]

In these equations size(T) is the size of a table in bytes. Functions time(q_i) and table(q_i) return the time it took to run a query q_i and a table that corresponds to query q_i.

In other words, the idea is to check that workload run times depend solely on the size of the table.

Having applied all the aforementioned measures, we have obtained the difference of about 10 – 15% in these values. We deemed such a result acceptable and decided to start evaluating the algorithms.

Finally, we must note that our matrix clustering algorithms are parallel [16]. However, in this paper we did not consider them and instead employed their sequential versions.

C. Experiments

In our study, we have addressed two applicability aspects of matrix clustering algorithms: quality of generated partitioning schemes and algorithm run times. Both of them are important since quality is the primary characteristic of any partitioning algorithm, and run times determine its suitability for on-line vertical partitioning.

To evaluate the quality of partitioning, we have compared algorithm A11 to the best of other matrix clustering algorithms (according to our previous studies [14], [15]) — A09. This algorithm has three different strategies that were described earlier. In our experiments we compare the quality of resulting partitions of all three of them with the ones obtained by A11.

To conduct experiments we have employed the “Star” table of the SDSS (Sloan Digital Sky Survey) dataset. The SDSS is a publicly available astronomical database that contains detailed three-dimensional maps of the Universe. It is frequently used as a testing dataset in various data partitioning studies. We have used the following pack: SDSS-IV Data Release 14, 2016. Its “Star” table contains 509 attributes and 492515 records.

To obtain representative workloads, we have also used the SDSS dataset. In SDSS, it is possible to see what queries users have issued via a special website1. Using this website, we have selected 8 queries from the workload that address solely this table.

In our first experiment we have varied the cohesion measure (a ratio of 1 in the resulting matrices) for three strategies of A09 and compared it with A11. The results are presented in Figure 5a. On this chart, each bar represents the performance of an individual algorithm with the corresponding strategy. There also are two horizontal lines: not clustered and pinched not clustered. The first one is the workload run time on the original, unmodified table. The second is the workload run time on the cleaned up original table, containing only 30 attributes that are referenced in the workload. In this experiment we varied the cohesion measure parameter.

To evaluate algorithm run times we used both SDSS and synthetic (generated) tests. The results of the SDSS tests are presented in Figure 5b. Here, we also vary cohesion for the same four algorithms.

In the synthetic tests, we have tried to study the scalability of the A11 algorithm in terms of run times. For this, we have generated a set of random 0-1 matrices with different probabilities of having 1 in each position (cohesion). Then, we have examined the dependency of the run time on the size of the matrix. The specified threshold was set to 0.9 in all experiments. If the threshold is more than the used cohesion, then a solution (the original matrix) is found almost

1http://skyserver.sdss.org/log/en/traffic/
immediately. We also set a time limit of 2 hours, after reaching which the algorithm is stopped.

We started with square matrices (see Figure 4a), then separately evaluated the influence of the number of columns (Figure 4b) and the number of rows (Figure 4c) on the algorithm run time. In the last two experiments we fixed one dimension to 20 and increased the other up until the time limit was reached.

Finally, we have looked into the storage requirements of these algorithms (Figure 6). Here, we show the required disk space for each generated configuration. On top of each bar, an overall number of fragments is shown. We have also divided each bar into parts representing the sizes of resulting fragments. The sizes of original and pinched tables are shown by horizontal lines.

V. RESULTS AND DISCUSSION

• All of the algorithms produced partitioning schemes that provide better performance than the original and pinched tables, regardless of the cohesion value.

• The quality of produced solutions heavily depends on the cohesion value. Starting with the cohesion value of 0.8 results of A11 start to rival the results of the best A09 strategies. However, up to this point, the clear winner is A09 with replication.

• Overall, the best result was produced by a replicating variant of A09 (3.358, cohesion=0.55), with a separate variant of A09 being the fourth (3.500, cohesion=0.8), and A11 being the fifth (3.553, cohesion=0.8).

• It is interesting to note that there is some sort of a global minimum at the 0.7 point. Here, the total time over all algorithms is minimal in the whole cohesion range.

• With the SDSS workload algorithm A11 works almost ten times faster than A09, regardless of the employed strategy. Note that increasing the target threshold also increases run times. For A09, run times increased from less than 1 second to almost 140 seconds, while A11 took 0.06 and 0.119 seconds respectively.

• The scalability of A11 is not as good as desired. However, two points should be taken into account. Firstly, run times depend on the number of referenced attributes in the workload, not on the total number. Secondly, in our scalability experiments we used an extremely large threshold of the cohesion — 0.9. Finally, the author [20] noted that it is possible to interrupt the algorithm earlier while still obtaining decent results. Therefore, further studies are needed.

• Increasing the number of attributes impacts run times more than increasing the number of queries. In two hours time it is possible to process either a 20 × 25 matrix or a 205 × 20 one.

• The solutions produced by all algorithms require from 1.5 to 2 times more disk space than the pinched table. Increasing the target threshold increases the number of fragments and the overall required disk space. Interestingly, for high cohesion values A11 produces more fragments, but does not help to improve performance.

VI. THREATS TO VALIDITY

We have identified a number of issues that should be kept in mind while discussing our results:

1) First of all, the policy of database restarts after each query may be unfair. In real-life scenarios where these algorithms will hypothetically be used, database caching would be present. However, such scenarios are nearly impossible to simulate since they require hundreds or thousands of real queries and more important, their frequencies and arrival patterns.

2) Next, the SDSS dataset is only a single dataset, so the results may differ on other datasets. Moreover, it is a scientific dataset used by the astronomy research community and therefore, its queries and data may not be comparable to the industrial ones. Nevertheless, it is popular in the vertical partitioning community (e.g. see [33]–[36]) due to the lack of industrial schema-less benchmarks.

3) There may be errors in our implementation of these algorithms. In order to mitigate this threat we have tested our implementation on example matrices presented in the considered papers and ensured that the resulting partitioned matrices are the same. Furthermore, to address this issue we plan to release the source code on GitHub.

4) Contemporary DBMSes are very complex systems in which minimal changes to inputs may drastically affect performance. Therefore, during experimental evaluation performance may change not due to vertical partitioning, but due to other events, such as query optimizer selecting a completely different plan. To counter this we have carefully checked query execution plans to find and eliminate any inconsistencies. We have also devised a criterion that allows to detect such inconsistencies in simple cases.

5) We have considered a relatively simple workload which involves only a single table. Having to perform extra joins in addition to the partitioning-induced ones may significantly decrease overall performance and thus, the desirability of vertical partitioning. However, joins with other tables are extremely rarely considered in literature [3]: only a handful of studies address them.

VII. ACKNOWLEDGEMENTS

We would like to thank Anna Smirnova for contributing to the editing and proofreading process.

VIII. CONCLUSION

In this paper we have presented a quantitative study of two recent matrix clustering algorithms. We have studied their output quality, run times, and storage requirements using both synthetic and real datasets.

Our evaluation has shown that for schema-less data all algorithms can produce a beneficial configuration, while a
Fig. 4: Run times of the A11 matrix clustering algorithm, synthetic datasets.

(a) A11 run times on a square matrix  
(b) Dependency of A11 run times on matrix width  
(c) Dependency of A11 run times on matrix height

Fig. 5: Performance of the A11 and A09 matrix clustering algorithms, SDSS datasets.

(a) Quality of partitioning  
(b) Algorithm run times

45
replicating variant of A09 is 10% better than A11. However, A11 is significantly faster and more importantly, less impacted by the target threshold parameter.

REFERENCES


Designing a DBMS Development Course with Automatic Assignment Evaluation

Viacheslav Galaktionov\textsuperscript{1, 2}  
\textsuperscript{1} JetBrains Research  
\textsuperscript{2} Saint-Petersburg State University  
Saint-Petersburg, Russia  
viacheslav.galaktionov@gmail.com

George Chernyshev\textsuperscript{1, 2, 3}  
\textsuperscript{1} JetBrains Research,  
\textsuperscript{2} National Research University Higher School of Economics,  
\textsuperscript{3} Saint-Petersburg State University,  
Saint-Petersburg, Russia  
g.chernyshev@spbu.ru

Abstract—Due to the constantly growing amount of data in the world, we need better ways to process it. Conducting research and development in this area requires skilled workforce. Different universities provide different courses to prepare people for this line of work.

In this paper we present our approach to conducting practice sessions within a DBMS development course. We describe some of the approaches implemented by other universities, outlining their advantages and disadvantages. A popular approach is to provide students with a prototype of some DBMS and let them incrementally improve it by completing certain tasks. The two most important problems in these courses are 1) choosing a DBMS (an industrial or educational one), within which students should work; 2) deciding whether to employ an automated testing system, and, if so, which one. In both cases we take a look at several options and justify the necessity to create a new one, which we then describe. In total, we have developed the following: a base prototype of a row-store query executor, an automated testing system, a set of problems along with reference solutions and test cases. Finally, we present the results of a test run involving 17 undergraduate students.

Index Terms—education courses, education, databases, query engine, query processing, database internals

I. INTRODUCTION

It is well-known that the amount of data that needs to be processed is increasing with an unprecedented speed \cite{1}. This is mostly related to the emergence of such areas as Big Data, the Internet of Things and cloud computing. The research of existing data storage and processing methods and the development of new ones are thus becoming more and more important.

Naturally, conducting said research and development requires a great amount of highly-qualified workforce. Preparing such cadres is an important task that perhaps all universities attempt to accomplish.

There is a multitude of courses aimed at improving qualifications in subjects related to databases. They can be divided into two categories:

1) Introductory courses, that explain a specific set of basic terms. Usually such courses consider the classic relational model and teach the students to apply it, but sometimes they can include information on various NoSQL systems.

2) Advanced courses, whose main task is to teach students to actually develop DBMSes. A lot of attention is paid to the internals of one or several classes of systems, as well as the most important algorithms. Student either develop their own system from scratch or modify an existing one.

Evidently, these two categories serve different purposes and as such have to be taught differently. Let us concentrate on the advanced courses in this paper. A question arises: how should one organize such courses? Clearly, just giving lectures to the student will not be enough because of the practical nature of the covered topics. The students need to be given an opportunity to apply their new knowledge in order for them to fully understand the material. This means that special attention should be given to practice sessions. In this paper we present our approach to conducting practice sessions within a DBMS development course.

The contribution of this paper is the following:

1) An overview of some of the approaches to conducting practice sessions within advanced database courses used in different universities.

2) The structure of our approach: the overall idea of the course, the used DBMS prototype, the task set, approach to testing students’ solutions.

3) The results of the test run of our course, a description of our experience and the encountered issues.

This paper is structured as follows. In Section II we describe various DBMS development courses. Next, in Section III we discuss overall architecture of our approach, and in Section IV we enumerate various security measures that we undertook. The syllabus of our course and the idea of proposed tasks is described in Section V. The Section VI presents the outcome of the first test run, justifies the benefits of our approach and describes the encountered issues. The future work and conclusion are presented in Sections VII and VIII, respectively.

II. RELATED WORK

Conducting advanced database courses is not a new problem, there are publications describing experience of many universities \cite{2–6}. The referenced papers provide two different viewpoints on how such a course should be organised:
Students of the course described in [2] had to modify PostgreSQL, a DBMS used in the industry. However, due to complexity of PostgreSQL’s architecture, only two of the tasks required students to actually modify its code.

On the other hand, the course described in [3] employed a DBMS developed specifically for it, SimpleDB. It was made with code clarity in mind, sacrificing performance where necessary. This allowed students, who were new to the subject, to find their bearings in the code and start modifying it. Because of this, the number of programming-related tasks in this course was 9.

A similar approach is being used at Harvard [4] right now. There, students implement their own main-memory column-store. They cover such topics as indexing methods optimized for main-memory and shared scan methods. During the class hours students discuss state of the art research papers.

In Russia such courses also exist. In 2004 the South Ural State University offered a course “Parallel database systems” [7], where students had to develop their own prototype of a parallel database management system using the MPI standard. In the Computer Science Center [5] course “Software engineering for big data” students were offered to implement a distributed key-value data store and an application. The assignment encouraged team participation and there were 4 tasks overall. Several years ago Innopolis Univeristy also offered [6] a three-week assignment for building a simplified relational query engine in Python. This project was aimed for team participation also. To the best of our knowledge, both these courses involved manual checking of the solutions.

The second approach appears to be more desirable, as it is both easier for the students and more saturated, i.e. instead of simply reading about different algorithms and approaches, the students will have to actually implement them.

Another advantage of the second approach is that the students are given an opportunity to improve their skills related to systems programming as well as complex system development [8], [9].

Because of the aforementioned reasons we have decided to take the second approach. However, that raises the question: which DBMS should we use? SimpleDB itself is hardly an option, since it puts a lot of attention on multi-user operation, which is not very interesting for us but complicates the code somewhat.

We have considered other systems as well: Minibase [10] and MinSQL [11]. The former provides a great set of features and comes in two version: Microbase, which is freely available to everyone but has a heavily restricted feature set, and Minibase, the full version, which is available only to teachers. However, the source code for the full version has already been published by third parties, which makes it easy for students to cheat. The latter, MinSQL, has never been made public.

Therefore, we have decided to develop our own educational DBMS.

Another important problem is grading students’ work. In the case of complex systems like DBMSes it becomes too difficult to assess the code by just reading it. Some form of automated testing is required. Trusting the student to write their own tests is not an option, and handing out tests to students could lead to them writing code that’s designed to pass the tests instead of actually solving the problems.

A more correct and modern approach is to allow the students to upload their code to some testing system, which runs various tests on it. Besides testing for correctness and performance, such a system can recognize different kinds of cheating: copying others’ solutions, DoS attack, etc.

This approach is very popular with programming contests and online courses. In both cases there is a stream of solutions that is too large for a group of people to evaluate in a reasonable time period.

Of course, automated testing is used outside of these areas as well. Code quality is an important characteristic of any software, so it receives a lot of attention. There are industrial systems for automated code testing.

We have evaluated multiple systems from different areas. Let us summarize our findings:

1) **Programming contest platforms.** We have considered Yandex.Contest [12] and Codeforces [13], which are the most popular platforms in Russia. These systems are capable of testing the code for correctness, performance, and are also protected against DoS attacks. However, they are expect the users to provide a single source file. This becomes a problem with complex systems like DBMSes. It is possible to put the entirety of source code into one file, however, that is not something we want to teach our students.

2) **Online course platforms.** This area was represented in our research by Stepik [14] and Coursera [15]. They exhibit the same problem as the programming contest platforms. However, they have additional disadvantages. For example, to the best of our knowledge, it is impossible to create a private Coursera course. Furthermore,

3) **Industrial testing systems.** We have looked at Travis CI [16] and Jenkins [17]. While they provide exceptional testing capabilities, they are not well-suited to track students’ progress.

Due to the lack of a system that would meet all our requirements, we have decided to implement our own.

### III. Testing System Architecture

Our system is organized as a set of Docker containers. The system provides a web interface, where the students can submit their code for testing and the teacher can track their progress. The main parts of the system are as follows:

1) **Web server.** The system has two kinds of user accounts: for students and for teachers. The interface is structured

---

1. [https://github.com/alesavin/csc-bdse](https://github.com/alesavin/csc-bdse)
Figure 1. Interaction between different parts of the system
differently depending on who the user is. The student
will see a list of tasks, some of which will be marked as
completed if they have submitted a solution that passed
all tests successfully. The teacher can add and remove
students from the course, make various changes to the
problem set, and track the students’ progress.

2) Database. We use PostgreSQL to store the information
on students, tasks, and submissions. We also use it to
organize the testing queue, which will be described later.

3) Testing queue. This module acts as a mediator between
the web server and the testing container. It is also
responsible for storing the uploaded solutions on disk.
The testing queue provides an HTTP API that is used
by the web server whenever a student uploads a new
solution or whenever someone wants to download a
previously submitted solution.

4) Testing container. Completely isolated from the rest
of the network, this module encapsulates building and
running the code. The entrypoint of that container is a
script that will unpack and compile the code and then
run our testing program. This program will run a set of
queries and check their results, meanwhile providing a
log that can help the students understand how and why
their code failed. However, this log is very concise and
does not provide the student with all information about
errors in order to keep the test data secret.

The diagram of interaction between different parts of the
system is presented in Figure 1. Note that only the web server
has access to the Internet.

The testing system was written in the Go programming
language. The net/http and html/template packages from its
standard library were used to implement the web interface
and the testing queue logic.

Now that we’ve covered the structure of our system, let us
describe how it operates when a student submits a solution.
First of all, the web server will receive the code archive and
pass it to the testing queue via its API. Then, the testing queue
will save the archive on disk, and then add two records to the
database: one that describes the submission itself, and one that
describes the testing result.

The testing queue process contains a set of coroutines, each
of which periodically queries the database for submissions
that need to be tested. If there is a solution that is ready to
be tested, the coroutine will lock the corresponding
database record, and start a testing container. This container
will be limited in its resources such as the amount of RAM or
execution time. These limits are configurable via the teacher’s
web interface on a per-problem basis.

Once the testing container stops working, the coroutine will
determine why it stopped and assign the submission one of
the following grades: accepted, invalid answer, runtime error,
timeout, compilation error. After that both the grade and the
log are saved in the database and the testing queue starts
waiting for a new task.

This process is depicted in Figure 2. Solid arrows represent
the path that a student’s code archive follows within our
testing system when it is uploaded. Dash arrows represent the
behaviour of a coroutine that is responsible for checking this
solution.

IV. Security Measures

Any user facing system should be sufficiently protected
from different kinds of attacks. Of course, our system is no
exception. A lot of attention has been given to protecting the
system from malicious behaviour that can be exhibited by
some students. Let us go over some of the precautions taken:
1) Our system works only over HTTPS. This helps us prevent traffic-sniffing in order to steal passwords, for example.
2) Testing containers are limited in their resources: RAM, the number of PID's, disk space. These limits help us prevent fork-bombs and memory floods.
3) By leveraging access rights we prevent the student's code from being able to write into the log generated by the testing program, thus preventing data leaks.
4) To prevent data loss, backup copies of all essential files were made daily.

V. Syllabus

The course was designed with undergraduate students in mind. Thus, it does not require any specific knowledge on the students' part: they should be experienced programmers with a basic understanding of the C++ programming language and be familiar with the UNIX environment. They should also understand what a database management system, a relational data model, and SQL are.

At the beginning of the course, all students are provided a prototype of a relational DBMS with minimal functionality:

- parsing queries written in a subset of SQL;
- a small set of physical operators: data source, filtering, and nested-loop join;
- building a simple query plan that can have at most one join operation;
- reading table data and printing query results in the CSV format.

The prototype is written in the C++ programming language. Architecture-wise it is a row-store that follows the Volcano query processing model. It was provided to the students through a GitHub repository [18].

A total of 8 tasks have been prepared, each of them aimed at expanding the prototype's functionality. The main topics were query optimization and execution with read-only workloads.

Along with the tasks we have written a reference solution to each task, using the simplest approaches. This reference solution was used during test generation to keep track of the expected execution time.

At the end of the course students who have completed all tasks will have developed a DBMS with the following features:

- block-oriented data processing;
- a larger set of physical operators: cross product, merge join and hash join, projection, multiple implementations of duplicate removal;
- an optimizer that can optimally select physical operators and the order of joins given the available RAM amount;
- a rewriter able to simplify the predicates and recognize the inconsistent ones.

The students were supposed to work on their own. Weekly seminars were held so that they could consult the instructor regarding task formulations and any technical problems they encountered. In order to check that there was no cheating, we froze the submission three days before the final exam to look through the code.

VI. Course Test Run

This course has been conducted at the Higher School of Economics in Saint Petersburg for a group of 17 students. All of them were taught this course in the same manner, there was no division into experimental and control groups. The test run was approved by the administration of the St. Petersburg School of Mathematics, Physics and Computer Science. No personal information was ever retrieved, and students' behavior within the course was not matched to students' university record or any other personal data. The testing system was deployed on the server provided to us by the institution.

A. Quantitative analysis

First of all, let us estimate various metrics related to all components of our system. They are presented in Table I. Here, we try to show the effort it took to develop each of the components in lines of code (all involved programming languages combined) and man-hours. The first line describes testing system itself, which was the largest in terms of lines of code. However, in terms of man-hours, the development of test cases and more importantly calculating their time limits were significantly more demanding.

The next three rows describe the properties of the prototype that was handed out to students, our reference implementation, and an averaged solution. Please note that the reference solution was built on top of the base prototype. The last line describes our effort to conduct a source code post-review to check for cheating and other possible problems.

Our estimates (from previous years, where manual checking was performed) show that each task requires at least two attempts, where each attempt lasts about 10 minutes. Therefore, the overall time required to check all problems that are offered in our new course would be approximately 10 minutes \(\times 2 \times 8 \times 17 = 45\) hours. Each practice class session lasts 90 minutes (once a week) and there are \(\approx 14\) such sessions in a semester, which gives us 21 hours in total. Therefore, it would be impossible to check all these solutions without involving additional reviewers.

Moreover, this number of additional man-hours required to prepare this course has been compensated by the following:

1) **Flexibility.** Students can check their solutions at any arbitrary time and can work at their own pace. Further-
more, they can attend class sessions only if they have questions.

2) Quality. We have improved the quality of the course by introducing precooked test queries and answers. This removes the possibility of instructors forgetting to check some particular test cases.

3) Reusability. The testing framework we have developed can be reused during the next iterations of the course.

4) Scalability. Using an automated testing system makes it significantly easier to increase the number of students in the future runs.

B. Qualitative analysis

The course has received good feedback from the students. However, we have found some problems in the way our course was organized:

• We have simplified the code for tuples too much, so in order to make block-oriented processing beneficial, the students would have to rework a significant part of the system, especially since this task was given late in the course.

• Easy access to automated testing prevented students from writing their own tests. This has reflected on the total number of submissions, which has exceeded 1000. Perhaps we should have provided students with a data generator instead of relying on them to find or develop one.

• Testing correctness by running a query and checking its result can be excessive, especially when testing such features as query rewriting. Furthermore, writing tests becomes unnecessarily hard in such cases, since the only criterion for success is whether the queries can be executed within a given time limit. A better approach would be to check the query plan directly.

• At the earlier stages of the course (before implementing a full-blown query optimizer) query execution time depended heavily on the order of joins. This made it impossible for one of the students to pass the tests despite having a perfectly working solution.

• Due to the fact that each run of the students’ programs corresponded to running a single query, some students made questionable design choices. For example, two students updated catalog information during query rewriting instead of relying on temporary data structures.

• There are some things that cannot be tested in an automated manner. The greatest example of that is student’s understanding of the code they have written. There was an exam at the end of the course, however, it was theoretical in nature and therefore there was no place for code-related questions. Conducting code-review sessions during practice sessions would be greatly beneficial.

VII. Future Work

This was just the first iteration of the course. We hope to improve our testing system and additional materials. In addition to fixing the problems mentioned in Section VI, we plan to do the following:

• Improve the user interface of our testing system, both in terms of functionality and visual design. As an example of missing functionality, the system might benefit from having an option to let the students ask questions so that the teaching assistants could answer them.

• Transfer the code submission process from an archive-based one to a git-based one. We expect that to bring multiple improvements. First of all, the process should become easier for students. Secondly, experience shows that many students will use git anyway. We have noticed that almost a half of our students have made their own publicly available forks of our GitHub repository. These forks included their solutions to the problems, which is undesirable, as it makes it easy for future students to cheat.

• Add means to check if a solution was copied from some other student. This time we had to dedicate three days before the exam in order to manually check all submissions that were accepted by the testing system.

• Make building and running the code separate isolated stages. This would allow us to keep full compilation logs and show them to students in case their code fails to build, as well as measure the actual query processing time. The latter would in turn allow us to range the students based on their code’s performance, thus giving them an incentive to find better solutions.

• Provide students with tools to test their code locally: a data generator; some simple logging facility; and a verbose version of our testing program, which would give the user detailed error messages without being afraid to leak test data.

VIII. Conclusion

In this paper we have described our experience of organizing practice lessons for a DBMS development course. We have outlined some of the approaches that have been taken by others before us and outlined their advantages and disadvantages.

In order to conduct the course we have developed a prototype of a simple DBMS and put together a set of problems for the students, along with a reference solution. To simplify the testing process for all parties involved, we have developed a system for automated testing.

We also describe the problems we have encountered during this course. Most of them have to do with the fact that this was the first iteration of this course and we lacked the time to plan and implement every desirable feature.

However, we believe our approach is viable and therefore intend to continue conducting courses in this manner. There is much to improve in our course besides dealing with the aforementioned problems. Our plans for future work are also described in the paper.

REFERENCES


Abstract—Existing word embedding models represent each word with two real-valued vectors: central and context. This happens because of words relations asymmetric nature and requires more time and data for training. We introduce a new approach based on asymmetric relations that uses the advantages of global vectors model. Due to the reduction of asymmetric information impact on resulting words representations, our model converges faster and outperforms existing models on words analogies tasks.

Index Terms—SSDE, word embedding, matrix decomposition

I. INTRODUCTION

Understanding words relations in the context of natural language is an easy task for human but not for computer. We need to teach computers how words are related and what meanings they have, depending on the context. To make it possible for a machine to process words, they have to be presented in digitized format. This leads to the idea of real-valued vector representations — word embeddings.

Most works on word embeddings focus their attention on preserving two words properties in their representations. The first property is that words relations and similarities can be described using distances and angles between word vectors. For example, closer-further feature: “yellow” is closer to “red” than to “smart”. In vector form it can be presented as

\[
\text{Yellow} - \text{Red} < \text{Yellow} - \text{Smart}
\]

This property is widely used for synonyms search. Another property is words analogies. The corresponding feature was introduced by Mikolov et al. [1], designed to learn words similarities. For example, “Paris” and “France” has the same connection as “Budapest” and “Hungary”. In vectors we can present it as

\[
\text{France} - \text{Paris} = \text{Hungary} - \text{Budapest}.
\]

This approach benefits models creating meaning based word vectors, while the closer-further feature is more practical and can be applied to clustering and classification tasks.

Word embeddings were originally created to be used in Natural Language Processing tasks. For example, one of the feature extraction techniques used for document indexing is latent semantic indexing [2]. Latent semantic indexing is a precursor for word embeddings embodying the same principles and ideas. Another task is sentiment analysis. One of the solutions for this problem is SentProp framework [3], it combines label propagation method with word embeddings to learn sentiment lexicons on domain-specific corpora. Another way to solve some of the Natural Language Processing tasks are Language Models. Nowadays state of the art decisions for Language Modeling are ELMo [4] and BERT [5]. Each of these methods uses prebuilt word embeddings as input data and can benefit from better embedding models. Therefore, creating better embedding models is still a relevant task.

There are three most popular and used word embedding models. Word2Vec is a local window-based method presented by Mikolov et al. [6]. It preserves words analogies feature, bringing closer vectors of words appearing in a similar context. Another approach is GloVe [7] which is trained on word-word co-occurrence counts. Authors noticed that to understand the relation of two words you can examine the ratio of their co-occurrence probabilities with various probe words, thus deploying words analogies feature. Third model – FastText [8] – is focused on distances/angles property. FastText uses character n-grams to enrich word vectors with subword information. This approach allows to use morphology information, therefore, choosing better vectors for sparse words and makes it possible to learn something for non-vocabulary words.

Words relations are often asymmetrical. For example, "New York" is a common combination of words meaning the name of the city in the USA. However, "York New" is a quite rare combination and does not mean anything specific. In all mentioned models words interaction is expressed in terms of the dot product of their vectors, that leads to a generation of two vectors for each word: central and context. For that reason, twice more parameters should be computed and, consequently, more time is required for learning. To solve this problem asymmetrical relations between word representation can be used instead of central and context vectors dot product.

In this work, we propose a Symmetric Skew-symmetric
Decomposition based model. We demonstrate that our method outperforms GloVe approach on its words analogies metrics.

II. RELATED WORK

There are many word embedding models known from the literature. But most of them were based on three principle approaches: Word2Vec [6], GloVe [7] and FastText [8]. All three models are widely used in language models and Natural Language Processing applications.

A. Word2Vec

Word2Vec is an approach introduced by Mikolov et al. [6] that preserves words analogies property. It suggests two language models: Skip-gram and CBOW. Both methods represent words relationships with the dot product of their vectors. As it was described in the introduction, relations can be asymmetrical, which leads to two vectors per word usage: central and context. Skip-gram and CBOW scan corpus with a sliding window. All words inside the window are considered to be in the same context, i.e. connected to each other. In both models all words inside one window get the same co-occurrence weight, i.e. are equal. We call this type of window "constant window".

Continuous Bag of Words (CBOW) is a model trained with "predict middle-word if you know surrounding context" task. The method tries to choose words central and context vectors, so that probability to predict the word in the middle of the sliding window, based on the rest of the window, would be high. The second model is called Skip-gram and is trained on the inverse problem: predict context with just one word in the center of the sliding window.

For each training step for each word, both methods should count the probability of using window middle-word in context with any other word from the vocabulary. It makes computational complexity too high. In later article [9] this problem was solved for Skip-gram model with Negative Sampling. Negative Sampling suggests counting the probability of middle-word being in the same context only with a constant number of positive and negative samples. Positive samples are words that often appear in one window with middle-word, they can be found before the training process. Negative samples are words that are unlikely to appear in context with middle-word. Mikolov et al. suggest getting negative samples from uniform distribution raised to 3/4rd power. This approach allows accelerating Skip-gram model calculations while being of the same quality.

Results of experiments have shown that Skip-gram method performs better on semantic tasks and their syntactic tasks results are very similar. Since Skip-gram can be trained easier than CBOW with same or even better results, later models use Skip-gram.

Skip-gram and CBOW models have several drawbacks. First, training time depends on the corpus size. Second, there are two vectors generated for each word, which requires more time and input data for training.

B. GloVe

GloVe model, for Global Vectors, suggested by Manning et al. also aims to preserve words analogies. The relationship of two words can be learned by examining their relations with other words. In this approach words relationships are represented with a matrix of their co-occurrences $X$, where $x_{ij}$ is how many times word $w_i$ was in the context with word $w_j$. This matrix should be constructed before the training process with one scan of the corpus. On each learning step we iterate through co-occurrences matrix and for each non-zero co-occurrence $x_{ij}$ calculate central and context vectors for corresponding $w_i$, according to value and direction of target function gradient.

In GloVe each word is presented with two vectors, similar to Word2Vec. A sliding window is also used to scan the corpus for co-occurrences matrix construction. Unlike the Word2Vec "constant" window, GloVe uses "shrinking" window. The weight of co-occurrence in the window linearly decreases with distance increasing. Authors did not explore how window type affects experiments results and did not give any details on such a choice.

C. FastText

FastText model, in contrast to Word2Vec and GloVe, was built to preserve words property of representing words relations in distances and angles between their vectors. This change allows the model to perform better on text classification tasks. Similar to two previous methods, FastText generates central and context vectors for each word and uses a sliding window to scan the corpus.

The main idea of this approach is to use character $n$-grams to build central vectors. During the vocabulary construction, each word is saved with it’s $n$-grams. For example, for the word “pencil" we also remember 3-grams "pe", "pen", "enc", "nci", "cil" and "il" in addition to the whole word sequence. 3-gram "pen" corresponding to the word "pencil" is different from the word “pen”. After that, during the training process, each sequence gets its own vector and resulting central vector is a sum of all $n$-gram vectors and whole word vector.

As it was mentioned, FastText has great results on text classification tasks but Word2Vec and GloVe outperform it on words analogies tasks.

III. THE SSDE MODEL

Words relations have asymmetric nature, for that reason all three approaches above generate two vectors for each word. The question is how to apply these central and context vectors. In GloVe, for example, there are several modes for what to use as a resulting vector. The default mode is a sum of central and context vectors. There was no intuition for this choice, although our experiments have
shown that the default mode indeed performs best. It is possible that Word2Vec, GloVe and FastText use more parameters than they really need, which means more time and input data is required for training. The subject of our research was to find out if words asymmetric information is really necessary to include into the resulting vector. To do that we introduce a Symmetric Skew-symmetric Decomposition Embedding (SSDE). It is based on GloVe model, mainly because it is faster than other existing models and performs better on word analogies metrics.

A. GloVe model analysis

The main idea of GloVe model: words $w_i$ and $w_j$ relation can be found by studying the ratio of their co-occurrence probabilities with various probe words $P(w_i, w_k)/P(w_j, w_k)$, where $w_k$ is a probe word. So, general model can be written as

$$F((u_i - u_j)^T v_k) = \frac{P(w_i, w_k)}{P(w_j, w_k)}.$$  \hspace{1cm} (1)

Authors say that due to exchangeability of words and context words function $F$ should be a homomorphism:

$$F((u_i - u_j)^T v_k) = \frac{F(u_i^T v_k)}{F(u_j^T v_k)}.$$  \hspace{1cm} (2)

This formula gives an idea that model $F$ is exponential, which in combination with Eqn. (1) leads to:

$$u_i^T v_k = \log P_{ik} = \log X_{ik} - \log X_i.$$  \hspace{1cm} (3)

After that GloVe brings biases to the formula. $\log X_i$ does not depend on probe word $k$ and is replaced with bias $b_i^v$. For word-context exchange asymmetry context bias $b_k^v$ is also included:

$$u_i^T v_k + b_i^v + b_k^v = \log X_{ik}.$$  \hspace{1cm} (4)

In this equation, right-hand side is what information model has to learn and left-hand side is how GloVe preserves it. This is optimized with weighted least squares notation and combining with the weighted least squares method will be very similar to GloVe target function:

$$\log \frac{p(w_i, w_j)}{p(w_i)p(w_j)} \Rightarrow e^{u_i^T v_j} \times \text{log} X_{ij} - b_i^v - b_j^v.$$  \hspace{1cm} (5)

Joint probability of words $w_i$ and $w_j$ are what in GloVe model is designed as co-occurrences matrix $X_{ij}$ and prior probabilities of words are designed as biases $b_i^v$ and $b_j^v$. In our experiments we tried both ways and obtained similar results for biases and probabilities usage. For that reason, we continued using prior probabilities in SSDE to decrease computational complexity.

B. Our model

From Eqn. (5) we see that GloVe represents words relations with dot product of their central and context vectors: $u_i^T v$. This is done to consider the asymmetry property that we want to remove. Central and context vectors dot product is equal to corresponding one-hot encoder vectors multiplication to central and context matrices product. Central and context matrices product can be considered as linear operator, and any linear operator can be decomposed to sum of symmetric and skew-symmetric matrices [10]:

$$u_i^T v_j = h_i U^T V h_j$$

$$L = U^T V = S + K.$$  \hspace{1cm} (6)

After that symmetric matrix $S$ (according to the property of symmetric matrices) can be written as a product of some low-rank matrix and its transpose. The same transformation can be used for the skew-symmetric matrix $K$ with multiplying lower-diagonal part to $-1$.

$$l_{ij} = s_{ij} + k_{ij} = a_i^T a_j + \xi_{ij} \cdot c_i^T c_j,$$  \hspace{1cm} (7)

The size of a matrix $A$ is $|V| \cdot l$ where $|V|$ - size of vocabulary, $l$ - word symmetric representation size. The size of a matrix $C$ is $|V| \cdot m$ where $m$ - word asymmetric representation size. Balancing between symmetric and skew-symmetric sizes we control the information distribution.
the way we need. For example, to reduce the influence of asymmetric information on resulting word representation we make constant $m$ much smaller than $l$.

In total, after rewriting GloVe target function (5) with Eqn. (10) and using the prior probabilities instead of biases, we get SSDE model target function:

$$Q = \sum_{i,j=1}^{[V]} f(p_{ij}) \cdot (a_i^T a_j + \xi_{ij} \cdot c_i^T c_j + \log p_i + \log p_j - \log p_{ij})^2, \quad (11)$$

- $p_{ij} = p(w_i, w_j)$ and $p_i = p(w_i)$ — are counted from the input corpus before the training process
- $\xi_{ij} = -1$, if $i > j$, otherwise $\xi_{ij} = 1$

On each training step we iterate through word-word co-occurrences matrix $X$. Each co-occurrence $x_{ij}$ shows how many times word $w_i$ was in the context with word $w_j$. We compute gradients for symmetric vectors and skew-symmetric vectors and update them according to the gradients.

Resulting word embeddings are vectors of symmetric matrix $A$. Since we wanted to remove asymmetric information influence on resulting word representations, vectors $c_i$ are only used for training. However, their properties worth further studying.

There are two ways to optimize function (11): 1) gradient descent, 2) stochastic gradient descent. The advantage of gradient descent is that it will eventually converge to better results. Though stochastic gradient has several methods that achieve reasonable results much faster than gradient-descent. Since we wanted to reduce training time, we decided to use Glove’s approach using adaptive gradient descent. GloVe authors also noticed that values slightly change on each stochastic gradient iteration which means computations can be done in parallel.

GloVe model shuffles whole co-occurrences matrix on each step of stochastic gradient descent.

$$\sum_{i=1}^{[V]} \sum_{j=1}^{[V]} f(X_{ij}) \cdot (a_i^T v_j + b_i^T + b_j^T - \log X_{ij})^2$$

$$= \mathbb{E}_{i,j \sim U(X)} f(X_{ij}) \cdot (a_i^T v_j + b_i^T + b_j^T - \log X_{ij})^2. \quad (12)$$

In SSDE model we shuffle only lines of co-occurrences matrix.

$$\sum_{i=1}^{[V]} \sum_{j=1}^{[V]} f(p_{ij}) \cdot (a_i^T a_j + \xi_{ij} \cdot c_i^T c_j + \log p_i + \log p_j - \log p_{ij})^2$$

$$= \mathbb{E}_a \sum_{j=1}^{[V]} f(p_{ij}) \cdot (a_i^T a_j + \xi_{ij} \cdot c_i^T c_j + \log p_i + \log p_j - \log p_{ij})^2. \quad (13)$$

Lines shuffle without columns shuffle makes computations cash-friendly, reducing cash-miss rate. This change allowed us to optimize model performance while quality remained the same.

IV. Experiments

A. Evaluation

To compare SSDE with GloVe we used metrics suggested in GloVe article. All the metrics are based on word analogies property. There are four words $w_1$, $w_2$, $w_3$, $w_4$, all associated with one topic and can be described as “$w_1$ is related to $w_2$ the same way $w_3$ is related to $w_4$”. This can be presented in vectors terms as $w_2 - w_1 = w_4 - w_3$.

According to the arithmetics law this can be rewritten as $w_2 - w_1 + w_3 = w_4(*)$.

Testing algorithm is: 1) get first three input words and count left part of (*) 2) among all vectors of our vocabulary find the closest vector $v$ to the previous step result (using cosine similarity) 3) if word corresponding to $v$ is equal to $w_4$, then this experiment was successful, otherwise it failed.

We do not provide a comparison with CBOW or Skip-gram models, but, as it is shown in the article [7], GloVe performs better than the other baselines.

‘Tab. I” shows all metrics that were used to evaluate both GloVe and SSDE models. Five of these metrics have semantic nature, for example,

“King” − ”Man” + ”Woman” = ”Queen”.

While the other nine are syntactic, for example,

“Dangerous” − ”Danger” + ”Beauty” = ”Beautiful”.

B. Results

We compared GloVe and SSDE models on corpus composed of 100 Mb of articles from English Wikipedia. For corpus scanning we used symmetric shrinking window of size 30. All models were trained up to convergence. Studying of the constant window and asymmetric window results will be completed in future work.

Tab. II shows the performance of GloVe and SSDE models with an equal number of parameters trained. Our approach significantly improves scores both for semantic and syntactic tasks.

Tab. III shows results of GloVe and SSDE models with equal sizes of word embeddings vectors. As it was mentioned, GloVe model uses a sum of central and context vectors as the resulting representation and SSDE model uses only a symmetric vector. Similar or even higher scores can be obtained with SSDE model with the same representation size as GloVe, but almost twice a smaller number of parameters.

All the results were obtained on Inter Core i7 processor, 8GB, DDR4 memory type.
Table I

GloVe word analogies metrics

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Example</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantic</td>
<td>Capital common countries</td>
<td>Greece to Athens as Iraq to Baghdad</td>
<td>506</td>
</tr>
<tr>
<td>Semantic</td>
<td>Capital world</td>
<td>Nigeria to Abuja as Ghana to Accra</td>
<td>4525</td>
</tr>
<tr>
<td>Semantic</td>
<td>City in state</td>
<td>Illinois to Chicago as Texas Houston</td>
<td>2467</td>
</tr>
<tr>
<td>Semantic</td>
<td>Currency</td>
<td>Dinar to Algeria as Kwanza to Angola</td>
<td>866</td>
</tr>
<tr>
<td>Semantic</td>
<td>Family</td>
<td>Brother to Boy as Sister to Girl</td>
<td>506</td>
</tr>
<tr>
<td>Syntactic</td>
<td>Adjective to adverb</td>
<td>Calm to calmly as Happy to Happily</td>
<td>992</td>
</tr>
<tr>
<td>Syntactic</td>
<td>Opposite</td>
<td>Aware to Unaware as Clear to Unclear</td>
<td>812</td>
</tr>
<tr>
<td>Syntactic</td>
<td>Comparative</td>
<td>Worse to Bad as Bigger to Big</td>
<td>1332</td>
</tr>
<tr>
<td>Syntactic</td>
<td>Superlative</td>
<td>Worst to Bad as Biggest to Big</td>
<td>1122</td>
</tr>
<tr>
<td>Syntactic</td>
<td>Present Participle</td>
<td>Coding to Code as Dancing to Dance</td>
<td>1056</td>
</tr>
<tr>
<td>Syntactic</td>
<td>Nationality Adjective</td>
<td>China to Chinese as Poland to Polish</td>
<td>1599</td>
</tr>
<tr>
<td>Syntactic</td>
<td>Past Tense</td>
<td>Danced to Dancing as Flew to Flying</td>
<td>1560</td>
</tr>
<tr>
<td>Syntactic</td>
<td>Plural</td>
<td>Bananas to Banana as Birds to Bird</td>
<td>1332</td>
</tr>
<tr>
<td>Syntactic</td>
<td>Plural Verbs</td>
<td>Eats to Eat as Says to Say</td>
<td>870</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>19545</td>
</tr>
</tbody>
</table>

Table II

Experiments with equal parameters number

<table>
<thead>
<tr>
<th>Model</th>
<th>Average sem score</th>
<th>Average synt score</th>
<th>Average total score</th>
<th>Sec. per iter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>GloVe-25</td>
<td>12.9%</td>
<td>12.3%</td>
<td>12.5%</td>
<td>13</td>
</tr>
<tr>
<td>SSDE-40-5</td>
<td>19.1%</td>
<td>22.5%</td>
<td>20.4%</td>
<td>11</td>
</tr>
<tr>
<td>SSDE-40-10</td>
<td>19.6%</td>
<td>22.6%</td>
<td>20.8%</td>
<td>12</td>
</tr>
<tr>
<td>GloVe-50</td>
<td>20.3%</td>
<td>29.4%</td>
<td>23.9%</td>
<td>18</td>
</tr>
<tr>
<td>SSDE-80-5</td>
<td>24.3%</td>
<td>39.8%</td>
<td>30.6%</td>
<td>17</td>
</tr>
<tr>
<td>SSDE-80-10</td>
<td>25.1%</td>
<td>39.7%</td>
<td>31.0%</td>
<td>18</td>
</tr>
<tr>
<td>SSDE-80-20</td>
<td>25.5%</td>
<td>41.4%</td>
<td>31.9%</td>
<td>20</td>
</tr>
<tr>
<td>GloVe-80</td>
<td>25.3%</td>
<td>40.6%</td>
<td>31.3%</td>
<td>24</td>
</tr>
<tr>
<td>SSDE-120-10</td>
<td>25.3%</td>
<td>48.1%</td>
<td>34.5%</td>
<td>25</td>
</tr>
<tr>
<td>SSDE-120-20</td>
<td>26.3%</td>
<td>47.8%</td>
<td>35.1%</td>
<td>27</td>
</tr>
</tbody>
</table>

a"GloVe-n" – vector size n.
b"SSDE-m-l" – symmetric size m, asymmetric size l.

We showed that it is possible to train high-quality word vectors using a little information on the asymmetry of relations, comparing to the popular word embedding model with highest scores on word analogies tasks – GloVe. Since our approach computes a twice smaller number of parameters, it requires less time to train the model.

We analyzed GloVe model and introduced a new model – SSDE – that combines the advantages of GloVe with our ideas on asymmetric relations. Comparison of SSDE with GloVe has shown that our model outperforms GloVe on word analogies metrics, while GloVe, according to the article [7], outperforms CBOW and Skip-gram models.

B. Future work

SSDE model, similar to GloVe and Word2Vec, uses a sliding window to scan the corpus. We assume that depending on the type of the window used, results may be different for metrics of different types. Constant windows might perform better on synonyms search tasks, while the shrinking window could be a good choice for word analogies tasks. So, in future work, we will examine window type influence on different metrics types.

Currently, we only use vectors with symmetric information for resulting word embeddings. However, there might be some interesting information encoded in asymmetric vectors. For example, L1-regularization turn most of the skew-symmetric vectors to zero. There might be some connection between those words which corresponding skew-symmetric vectors are not zero. In future work, we will study the asymmetric component of SSDE and analyze if there is any pattern that might increase performance on some tasks.

Window size and symmetry influence on model performance is another aspect that was not examined. Importance of asymmetric information might increase for highly asymmetric windows.

V. Conclusion

A. Achievements

In this paper, we studied the necessity of word relationships asymmetric information for word embeddings.
REFERENCES


59
Medical Images Research Framework

Sabrina Musatian
Saint Petersburg State University
Saint Petersburg, Russia
Email: sabrinamusatian@gmail.com

Alexander Lomakin
Saint Petersburg State University
Saint Petersburg, Russia
Email: alexander.lomakin@protonmail.com

Angelina Chizhova
Saint Petersburg State University
Saint Petersburg, Russia
Email: chilina4@gmail.com

Abstract—with a growing interest in medical research problems and the introduction of machine learning methods for solving those, a need in an environment for integrating modern solutions and algorithms into medical applications developed. The main goal of our research is to create medical images research framework (MIRF) as a solution for the above problem. MIRF is a free open-source platform for the development of medical tools with image processing. We created it to fill in the gap between innovative research with medical images and integrating it into real-world patients treatment workflow. Within a short time, a developer can create a rich medical tool, using MIRF's modular architecture and a set of included features. MIRF takes the responsibility of handling common functionality for medical images processing. The only thing required from the developer is integrating his functionality into a module and choosing which of the other MIRF’s features are needed in the app. MIRF platform will handle everything else. In this paper, we overview and compare existing applications for handling operations with medical images, as well as describing basic ideas and functionality behind our own MIRF framework.

1. Introduction

Over the past decade, many kinds of approaches for solving problems in the field of medical images were explored. Because of these researches, the scientific community can now rapidly open new and more challenging tasks. However, these studies should go beyond just algorithmic decisions related to diagnosis and treatment using CT (Computed Tomography) and MRI (Magnetic resonance imaging) images. Doctors require high-performance real-time software systems that can assist in the diagnosis's determination of the patient and solve various related tasks. Hence, it is necessary not only to develop highly efficient algorithms for medical images analysis but also to integrate them into a convenient environment in which many other instruments essential for physicians may be seamlessly used. A set of medical tasks share many of these tools, which means that these tools can be provided within a single platform. In this paper, we investigate existing software systems for medical images and introduce our own framework (MIRF) for medical diagnosis, simplifying the development of medical instruments. The objectives of this work are to create an extensible platform for the development of medical instruments and to show successful applications of this library on some real medical cases.

2. Existing systems for medical image processing

There are many open-source packages and software systems for working with medical images. Some of them are specifically dedicated for these purposes, others are adapted to be used for medical procedures.

Many of them comprise a set of instruments, dedicated to solving typical tasks, such as images pre-processing and analysis of the results – ITK [1], visualization – VTK [2], real-time pre-processing of images and video – OpenCV [3].

Others solve problems related to image analysis of certain organs or diseases. For example, brain images analysis (FreeSurfer [4], SPM [5] and others). The extension of such software systems for solving a wide range of tasks in medicine is quite complicated or even impossible since most often the architecture of such applications was written for solving a specific task and it may be hard to generalize these approaches.

There are also many general-purpose medical imaging applications. Such systems provide basic functionality for working with images. However, they cannot be expanded to address any specific tasks (for example, segmentation or finding features inherent in certain diseases). Such systems are: Ginkgo CAD [6] and ClearCanvas [7].

Another class of medical software form expandable medical applications that focus primarily on the final usage by the doctors. They already provide all the basic methods in an integrated user interface, for example, Slicer [8], Weasis [9] and OsiriX [10]. The last one is an expensive commercial product and is not available to a wide audience. Such applications can be expanded with specifically written plugins for these platforms. However, this approach does not give the developers enough flexibility to create and adjust their own systems and functionality.

The most generalized and flexible product for working with medical images is MITK [11] – an open-source framework for developing interactive medical software systems. MITK combines the algorithms presented in ITK [1]
with the visualization algorithms from the VTK library [2]. MITK also supplements the functionality of these two libraries with some unique features, allowing its users to create a variety of medical programs from a broad range of functions. While MITK is a cross–platform framework, some versions have not been supported for years. Because it is originally written in C++, it requires to be built separately for each platform. Moreover, the developers have to use a custom build procedure provided by MITK to create and add new modules.

In this paper, we introduce our own open–source medical images research framework (MIRF) as an alternative to existing software systems for medical applications development. MIRF is written in Kotlin programming language with a focus on enabling a smooth integration between modern research in medical imaging. With the Kotlin at its core, MIRF can be smoothly integrated into any projects with Java Environment. We pay close attention to the possibility of integration of artificial intelligence and various machine learning approaches for diagnosis and treatment of various diseases. This is because nowadays the most effective solutions for medical image analysis problems are solved using machine learning or deep learning algorithms [12].

3. MIRF architecture

3.1. Structure

MIRF framework is represented as a collection of generic modules for various tasks. These modules are divided into two global packages:

- Core – the minimum set of necessary modules for the correct operation of the MIRF framework. This package includes modules that are used for transferring data into the internal representation, communication between modules and creating data processing pipelines.
- Features – contains modules with core user functionality that are needed to facilitate development: mechanisms for accessing data storages, adapters for various medical data formats, various pre–processing filters and image analysis tools. Any custom modules should extend the capabilities of this package.

3.2. Pipeline

Execution of any workflows in MIRF is implemented with Pipes & Filters [13] approach. For these purposes, various data handlers should be used to stick individual blocks together.

In the framework, any computational logic must extend the Algorithm interface. The Algorithm is a handler class that, when invoked, changes only the data submitted to it at the input. The Algorithm does not invoke any third–party code associated with data processing. It does not save data and acts solely as a data handler. This approach provides opportunities for flexible creation of algorithms and organization of hierarchies.

Algorithm instances act as filters in our architecture. The Algorithm class is encapsulated by the PipelineBlock class, which is the main entity used to transfer data between algorithms. The communication between the blocks is based on the Observer pattern – after the block executes the algorithm, it informs all its listeners about the completion of the calculations. Some blocks may also be engaged in the aggregation of data for the following blocks or have another specific purpose (for example, they indicate the completion of calculations in the pipeline).

The core architecture of MIRF may be seen at figure 1.

3.3. Data representation

Any data in MIRF should be derived from an abstract class Data. The main task of this class is to take over the management of the metadata, namely the list of attributes (AttributeCollection class). Any class inherited from the Data class should be used only as a data storage object. Instances of Data class are passed through the MIRF pipeline and act as Pipes in our Pipes & Filters approach. This ensures the clarity of the entity's purpose within the framework.

![Figure 1. The Core Architecture of MIRF.](image)
3.4. Pipeline initialization

MIRF provides common blocks that may assemble custom pipelines and control the functions that the user wants to be executed on the provided data. The pipeline initialization can be done with a few simple steps:

```java
val pipe = Pipeline("Pipeline name")
// Creating the blocks
val firstBlock = PipelineBlock(    Block parameters
)
...Initialization of other blocks...
// Creating connections between blocks
firstBlock.dataReady +=
    secondBlock::inputReady
...Initialization of other connections...
// Setting the first block and input
pipe.rootBlock = firstBlock
pipe.run(Data)
```

4. Medical images representation

4.1. MedImage

There are several common medical images formats, for example, DICOM [14] or NIfTI [15]. For a unified workflow with these formats and applications of common analysis algorithms, we have implemented a general class for representing medical images in MIRF. MedImage is a class, that contains a list of attributes extracted by certain rules, depending on the source format and the pixel representation of the image. Thus, all algorithms for working with medical images work with the MedImage class, which allows the library user to reuse and extend the existing code.

4.2. DICOM

DICOM format is represented as a set of key-value items, and the image itself is also stored by key, as a value. All sets of keys for DICOM images are strictly defined and are used everywhere by the medical community. To read DICOM images, we considered several libraries for working with this format in Kotlin: ImageJ [16], DCM4CHE [17], and PixelMed [18]. While ImageJ supports the DICOM reading, it does not provide the functionality to output images with this format. DCM4CHE is a rich toolkit for working with DICOM images, it provides a lot of functions to work with those images, using medical servers. Because we don’t want to overwhelm our library with unnecessary dependencies, we made our final decision towards PixelMed, which supports reading, working with attributes and writing of DICOM images without complicated workflows such as in DCM4CHE. After reading the list of attributes for a DICOM image MIRF converts it to the MedImage class by creating an internal representation of the attributes and extracting an array of images from the original format.

4.3. NIfTI

Another popular type of medical images is NIfTI [15]. There are a few differences between DICOM and NIfTI file formats, such as the data they store and storage representations. For instance, NIfTI metadata does not include patients or hospital related information. It only stores the image and MRI settings metadata. Also, NIfTI stores a set of medical slices within one file (a set of medical images), while DICOM usually stores them as separate files. To enable NIfTI usage in our framework, we used ImageJ [16]. Then, similarly to DICOM images, we convert the information received from the NIfTI to our internal MedImage representation, to make it possible for the same algorithms to work with different file formats.

5. Unique features

5.1. Tensorflow models integration

Because modern researchers are often using deep learning techniques for solving various problems in medicine, we paid special attention to the possibility of integration of those approaches effortlessly within our framework. We started with the most commonly used deep learning frameworks such as Tensorflow [19] and Keras [20]. As a result, integrating Tensorflow models is possible within MIRF Tensorflow block. Since Tensorflow provides a Java API for working with its models, it was possible for us to create a block which may run the provided models. To run inference on the prepared Tensorflow model, the Tensorflow Block with the models parameters should be instantiated. It is sufficient to pass in the path to the saved model and the names of the input and output nodes.

Also, since Tensorflow package provides Keras interfaces, it is possible to integrate not only Tensorflow models but Keras as well.

To the best of our knowledge, no other software for creating medical applications, provide such integration within its core functionality. We believe that this feature is very important in the modern medical applications development because it completely encapsulates the integration of the complex artificial intelligence models in real medical applications and enables developers to focus on creating new algorithms in their preferred languages and environments.

To use Tensorflow API in C++ or Java developers have to specify the Graph of the model and define many fields before they can run it. However, MIRF users can set up the Tensorflow block within just a few lines:

```java
val tensorflowModel = TensorflowModel(    MODEL_NAME, INPUT_NODE_NAME,    OUTPUT_NODE_NAME, OUTPUT_DIMS )
val tensorflowModelRunner = AlgorithmHostBlock<Data, Data>(
    {    tensorflowModel.runModel(
```
Figure 2. The data flow diagram for the multiple sclerosis analysis pipeline. MIRF reads a set of DICOM images and loads lesion masks from a baseline set. The follow–up set of images is pre-processed and the segmentation masks are calculated. Then, MIRF compares the baseline and follow–up images and generates a report based on this data.

5.2. PDF reports generation

There are various types of medical documentation that doctors generate after the patients appointment. Those documents usually include CT and MRI images as additions to the final diagnosis paper and recommendation. The outline and contents of medical reports are strictly regulated by the government standards and they vary by different criteria, such as organs, diseases or medical procedures performed. Doctors have to fill in those reports manually or semi-automatically and include images into them. MIRF provides tools for creating these reports automatically, based on the results of the specified pipelines. MIRF generates the report in PDF format and has all the necessary images already included.

We use algorithm class implementation for this purpose. It generates a report in the form of PdfElementData from input data. The final report is then created by PdfElementsAccumulator class, which takes the sequence of PdfElementData as input and draws them on the document. We use IText 7.1.2 [21] as the main library for working with PDF format.

MIRF provides a set of primitive modules that may be included in the final PDF report. We currently support tables, images, and raw text. If the user needs other instances in his report, he may create his own implementation of PdfElementData and include it in the final report.

6. MIRF applications

6.1. Multiple sclerosis analysis

With our framework, developers may easily create custom pipelines for specific tasks. This automates many manual scenarios and it can bring new features, unused before,
to doctors workflows. We take Multiple sclerosis analysis as an example of such a workflow.

Multiple sclerosis is an immune–mediated disorder, affecting the central nervous system. Patients with this disease have multiple lesions in the brain. Such patients have to take MRI scans twice a year. Doctors are comparing scans over time and check the growing process of lesions in the brain. They generate the report about this.

We implemented an application, that generates MS reports based on the baseline and follow–up sets of scans. This procedure saves a lot of time for doctors and optimizes their work at several steps.

The data flow diagram for this pipeline may be seen at figure 2. First, it reads a set of DICOM images and loads lesion masks from a baseline set. The follow–up set of images is pre–processed and the segmentation masks are calculated. We use Tensorflow block to perform segmentation on the images. Then, MIRF compares the baseline and follow–up images and generates a report based on this data. An example report for the MS pipeline may be seen at figure 3.

With this application, the segmentation, comparison and report generation is performed automatically for the doctor.

6.2. Brain tumor analysis

Another example, that shares common functionality with MS analysis is the brain tumor segmentation and a report generation from the obtained information. With this case, we show how MIRF core functions may be used in various scenarios and optimize doctors workflows.

According to [12], brain tumor segmentations are performed either manually or semi–automatically, as well as there is no registered case of bringing the modern research for this problem into real clinical trials. The main information that can be inferred from such segmentation on the early stages of treatment is the tumor volume and its relative volume to the whole patients brain. Hence, these discoveries should be added to the final disease statement. These actions (analyzing MRI scans, calculating the volume and including this information in a report) are performed manually by the specialists. As part of the final tool for working with various medical images, this pipeline may be easily included in our framework. For the brain tumor segmentation, we take an implementation of the state of art solution of this problem [22]. The algorithm for segmentation is implemented using Tensorflow framework and may be integrated as a model file with our general purpose Tensorflow block, described above. It takes MRI brain images in NIfTI format [15] and creates a mask, indicating different types of tumor tissues, where they are present (figure 4). Since MRI images are represented as a set of slices, where voxels in the slice correspond to some particular volume, it is possible to calculate volume, based on the number of voxels. The information about this encoding is stored in a medical image metadata and depends
on the MRI machine settings. MIRF calculates the tumor volume based on the segmented mask. Using the initial brain images the whole brain volume may be determined and relativities between those volumes are deduced. Then, MIRF creates a complete report with this information, using PDF generating tools. The data workflow and the blocks used in this pipeline may be seen on figure 5.

This example shares such blocks as image reading, segmentation and report generation with MS analysis workflow. It demonstrates how the core MIRF blocks may facilitate very different workflows from the medical point of view.

### 6.3. Skin cancer detection on Android

Due to the fact that we develop our framework on Kotlin programming language, it is possible to create not only cross-platform desktop applications but also mobile. This enables developers to deploy the same pipelines and scenarios on a wide range of devices without rewriting any code. To show the benefits of this approach, we created an Android application for skin cancer detection. For image classification, we use an open-source implementation of one of the deep learning algorithms for this problem [23]. It implements the deep learning model with the Keras framework usage. Since it is possible to convert Keras models into pure Tensorflow models, we may use the Tensorflow integration block from MIRF to deploy this model. As a result, we may use the same pipeline as for the desktop app on the phone to detect skin cancer from the phones images. It is required from the developer to write custom layouts on Android for the GUI. We are planning to resolve this issue in the future, by creating a pre-defined library of such
graphical interfaces, so the development of these apps may be performed with more ease and automaticity.

To show the simplicity of our approach, we provide the pipeline code that is used on Android to run this example. It takes an image path as an input and generates a label showing whether the mole is benign or malignant.

```java
val pipe = Pipeline("Detect moles")
val assetsBlock =
    AlgorithmHostBlock<Data, AssetsData> (...algorithm parameters...)
val imageReader =
    AlgorithmHostBlock<AssetsData, BitmapRawImage>
    (...algorithm parameters...)
val tensorflowModelRunner =
    AlgorithmHostBlock<BitmapRawImage, ParametrizedData<Int>>
    (...algorithm parameters...)
val root = PipeStarter()
// Make connections
root.dataReady +=
    assetsBlock::inputReady
assetsBlock.dataReady +=
    imageReader::inputReady
imageReader.dataReady +=
    tensorflowModelRunner::inputReady
// Run
pipe.rootBlock = root
pipe.run(MirfData.empty)
```

7. Conclusion

In this paper, we introduced the Medical Images Research Framework for the development of complex medical applications with various types of medical images. We investigated the existent solutions in this area and argued why we created such a tool.

We introduced the basic overview of the proposed architecture and its benefits in this paper, as well as the unique features of our platform. We believe that our framework will help in mending the gap between innovative research made in medical images analysis and delivering it to the final users. As our research is still early in development, we have many plans for further integration, such as adding most commonly used features (scales, segmentation masks, zooming, working with patients data) and GUI for them. We also plan on creating a visual programming environment based on our framework, so creating medical apps would be possible for people with little programming experience.

Out project is publically available and may be found at https://github.com/MathAndMedLab/Medical-images-research-framework

References

Abstract—DSM platforms that are based on metamodeling typically use the two-metalevel approach. Disadvantages of this approach were taken into account when developing the REAL.NET platform, and deep metamodeling approach was chosen. The following article describes an experiment to create a “smart greenhouse” programming technology on the basis of REAL.NET. The experiment has shown the efficiency of the platform for quick creation of tools for the end user programming. The article describes the platform itself and deep metamodeling approach as well as “smart greenhouse” programming technology based on it.

Index Terms—Domain-specific modeling, visual languages, multi-level metamodeling

1. Introduction

The days when visual modeling was considered a new silver bullet are long gone, but visual modeling is still a viable tool for end-user programming. Non-technical people often don’t have time to study even a simple programming language, and in scenarios where some simple programming is needed visual languages can be very effective alternative. We believe that number of applications of visual languages for end-user programming will grow with the adoption of Internet of Things, blockchain smart contracts, simplification of mobile application and web application development and so on.

Growth of visual end-user programming technologies is hindered by a very high cost of tool development and usability issues. Adequate tools for visual languages are much harder to develop than textual IDEs due to complexity of graphical editing features required, and when a tool is needed only to perform a small set of tasks for a limited amount of users, costs of its development are much higher than expected benefits. To address this, a considerable amount of research was done on creating tools that simplify visual modeling tools development — Domain-Specific Modeling platforms (or DSM platforms).

The core of every DSM platform is its ability to declaratively specify a visual language and automatically provide tools like visual editors, code generators, model browsers and so on. Many different approaches and formalisms were developed for language specification, most of them are using metamodels — models (visual or textual) that specify a set of valid models that becomes a new language. Metamodel is itself a model created using dedicated metalanguage, much like Backus-Naur form describing the grammar of textual language. There are several existing metamodel architectures, most widely known being two-level architecture (with dedicated metalanguage that allows to specify needed visual language), used, for example in MOF (Meta-Object Facility, metalanguage with which UML is specified) and several popular DSM platforms, like Eclipse Modeling Project [1] and MetaEdit+ [2].

Limitations of two-level metamodeling architecture became apparent when UML 2 standard was discussed (see [3], [4]), and new metamodeling architectures have emerged with the aim to simplify language definition and improve tool support. For example, UML is not able to capture instance-of relation between classes and objects on a language level, as “Class” and “Object” are different instances of the same element of metalanguage and are not related to each other. So every UML tool needs to have some complicated custom code to maintain consistency of models that use classes and objects — for example, that every object in a model has its corresponding class somewhere. One of the most developed architectures that allows to solve such problems is deep metamodeling [4], [5]. Deep metamodeling proposes to consider entities of a model as classes and objects at the same time (and call them "clabjects"). Clabjects can be used in a model and at the same time be used as types for a lower-level model, for example, clabject “Class” may exist on UML class diagram and have its instances on UML object diagram. With such formalism an object can not exist without corresponding class, since it is its instance, and no special support is required to capture such relation in a visual modeling tool.

There are many publications about deep metamodeling (starting from [6] and including pivotal works [4], [5]) and there are some tools that use deep metamodeling as a metamodeling framework (for example, Melanee [7] and textual modeling tool MetaDepth [8]). But most of such tools are purely academic research projects or are supposed to be used by software engineering professionals, so whether deep metamodeling architecture is usable and
beneficial for end-user modeling tools development, remains open question. Our research group had recently created rather successful visual programming tool employing DSM platform with two-level metamodeling architecture (in particular, QReal[9] DSM platform was used to create TRIK Studio [10] educational tool). A fact that resulting tool has several thousands of active users most of which are children who can not program in textual languages, is a good indication of feasibility of two-level metamodeling, but we experienced several language design problems that we were not able to solve with two-level architecture (see [10]). We decided to develop a new DSM tool using deep metamodeling architecture and to create an end-user programming tool using it to compare our experience with two-level metamodeling and to gain some experimental evidence on applicability of deep metamodeling to real-world end-user programming. “Smart greenhouse” programming tool was selected as our goal because it is relatively small and simple domain, but can be easily extended to more general Internet of Things applications, and there already exist some visual programming tools (Node-RED1, for example) with which we can compare our results.

The remaining part of its article provides brief overview of deep metamodeling, followed by a brief overview of REAL.NET — our DSM platform that supports it. Then a visual language for “smart greenhouse” programming is introduced as an instance of REAL.NET metamodel hierarchy and tools for working with this language are described. Next we summarize and analyze our informal experience creating this tool, compare our results to related work and conclude the article. We believe that main contribution of this article is in reporting the experience of deep metamodeling usage for end-user programming tool development, which, we hope, can be meaningful contribution to empirical data related to metamodeling architectures, and as such helps to advance a knowledge about visual languages.

2. Deep Metamodeling

Deep metamodeling was first proposed in 2001 [4] as a basis of new UML 2.0 standard, but UML 2.0, released in 2005, still used two-level metamodeling architecture (and uses it by today). Later deep metamodeling received attention of domain-specific modeling research community, several tools using it were developed (i.e. [7], [8], [11]) and it sparked a wide interest and debates in related multi-level metamodeling techniques (see, for example, architecture based on “Powertype” pattern [12], comparison of different architectures in [13] and empirical study [14]).

Main idea of deep metamodeling is to consider entity in a model as a type and an object simultaneously. Each element of a model (node or edge) can be an instance of an element of some other model (which is then considered as metamodel) and a type for some elements in other models (so our model can be considered a metamodel related to them). Elements, attributes and various other elements have numeric attribute called “potency”, which denotes how many times given element or attribute can be instantiated. For example, classic two-level metamodeling architecture can be considered deep metamodeling where potency of each element is either 1 or 0. Another example is a hierarchy of simplified UML metamodel, UML class diagram and UML object diagram, illustrated on figure 1.

![Figure 1. An example of class-object model hierarchy](image)

Potency allows language creator to influence model structure several metalevels below language metamodel, which can be useful to connect user models with “instance-of” relations (like in our example with classes and objects). It is useful not only to explicitly express “instance-of”, but also to give a limited ability for an user to specify language elements. It seems to be rather unexpected ability for a visual modeling tool, but has some important use cases — for example, user-created subprograms can be considered as a new language element, instances of “Subprogram”, but able to have their own instances — program calls.

For visual languages an editor shall be able to work with any element on a model, i.e. be able to correctly draw it, provide the ability to edit its attributes and so on. For this, all elements shall expose a set of properties that are not specified by their metamodel, but are determined by capabilities of an editor. To uniformly handle this, orthogonal metamodeling was proposed in [5] as an addition to “pure” deep metamodeling framework. With orthogonal metamodeling, each element is an instance of some element in a metamodel (which is called ontological metamodel) and at the same time it is an instance of some element in other — linguistic — metamodel, which determines only properties related to representation of an element. See figure 2 as an example of UML models hierarchy with an addition of linguistic metamodel.

3. REAL.NET Overview

REAL.NET is an implementation of DSM platform that is able to support deep metamodeling. We started working on REAL.NET with the aim to provide a tool and a set of libraries for experimenting with visual languages based on .NET platform, as an alternative to Eclipse Modeling Project, which is widely used for visual language research.

Figure 2. Orthogonal model hierarchy example

now. Eclipse Modeling Project targets Java platform and is highly dependent on Eclipse infrastructure, but there are many .NET applications that could benefit from visual languages, so we decided to develop our own DSM platform from scratch.

Component diagram with an overview of REAL.NET architecture is presented on figure 3.

Main component of a system is repository. Repository is able to store models and perform operations on them, including operations that require knowledge of model semantics, for example, instantiation. Repository provides API for higher layers that allows them to work with models in high-level terms, for example, create elements that are instances of given type, having given properties. Repository encapsulates knowledge about metamodeling infrastructure and contains a hierarchy of predefined models which are needed to define that infrastructure and semantics. All data structures used by repository to store models are defined by those models themselves, so it is possible to automatically generate repository data structures using predefined models — a repository is self-defined in this sense.

Repository is used by two visual editors — one is based on WPF framework, other uses Windows Forms framework. WPF framework runs only on Windows but allows much nicer-looking GUI, Windows Forms framework, despite its name, runs nicely on Linux and Mac OS. Right now we consider WPF editor as our primary editor. WPF editor consists of reusable controls library which provides components like palette, scene, property editors and so on, and a frontend which puts these components together. We actually have several frontends — one for general-purpose diagram editing, others — for a specific programming tools based on REAL.NET, for example, quadcopter programming environment.

It is possible to construct domain-specific tool using components from controls library, but there is another possibility — define a plugin which will be dynamically loaded by plugin manager and is able to add its capabilities to an editor. Plugins are much simpler to implement than custom editor, so for scenarios where GUI is not important (e.g. code generators) plugins are preferred way to provide domain-specific functionality.

REAL.NET uses hierarchy of metamodels to define its metamodeling capabilities, and the “smart greenhouse” language is an instance of Infrastructure Metamodel, which is itself an instance of Language Metamodel, which is an instance of Core Metamodel, which is an instance of itself. This architecture allows us to relatively easily replace any metamodel layer and implement different metamodeling architectures.

4. REAL.NET Metamodels Hierarchy

The base metamodel in REAL.NET is Core Metamodel. The key element of metamodel is “Node” and it is the instance of itself, while the other elements that are defined in this model, such as “Element”, “Edge”, “Generalization”, “Association” and “String” are the other instances of “Node” (“instance of” relation is itself modeled as “class” link from “Element” to itself). At the same time, all elements are inheritors of the “Element” and therefore should have an association relationship named “class” with some “element” inheritor. Full Core Metamodel is presented in figure 4, dashed lines represent “instance of” relations.

Figure 3. REAL.NET high-level architecture

Figure 4. REAL.NET Core metamodel

The following level is the Language Metamodel. It defines the elements with which the Infrastructure Metamodel
is built and contains edges, nodes and also definitions for “source”/“target” of edge and an “enum” type. The next level is Infrastructure Metamodel. It defines metamodeling capabilities that are used to create actual domain-specific languages, for example, this is the first model where “Attribute” notion is properly introduced. Infrastructure Metamodel models an interface between repository and editors and enables high-level operations such as creating a new model, creating elements and so on.

5. Visual Language

“Smart greenhouse” language is based on Infrastructure Metamodel and enables the creation of rules for autonomous greenhouse work using visual primitives. Consider two use case examples, when and to whom autonomous greenhouse work may be useful.

- Alice lives outside the city and has several greenhouses with different plant cultures. She spends a lot of time on opening/closing greenhouses every morning/evening when it gets warmer/colder. Therefore, she wants to automate this process, indicating in the scenario at what temperature windows should be opened in a specific greenhouse.
- Bob lives in the city, but has a greenhouse outside the city and has the opportunity to go there only on weekends. He wants to grow plants that require daily watering. Therefore, he needs the ability to set scenarios for watering plants depending on soil moisture.

Thus, the greenhouse scenarios should indicate which external conditions should trigger the device operation. Greenhouse program should react on the data that comes from sensors and send commands to actuators. So the paradigm of dataflow programming is useful to make the work more clear for end-users. Data flow is represented in form of edges that connect modules.

Greenhouse metamodel elements are shown on figure 5. There is only one association, it represents data flow in form of oriented edges that connect data sources with receivers. Source and target vertices should be descendants of the metamodel abstract node which is the instance of the infrastructure node. Since the entire system processes information from sensors and issues commands to the actuators, sensors are always sources and actuators are target nodes. The metamodel includes two types of sensors and three types of actuators. Each of them has port number of a physical device they introduce as an attribute. The metamodel abstract node also has logical operation and interval as descendants. In turn, the logical operation node has “AND” and “OR” operations as its descendants. The interval node has attributes of minimum and maximum values and sets an open interval for checking whether it has value that is passed to it. “Null” value of this attributes is interpreted as the absence of the upper/lower limit of the interval.

Each metamodel element also has the following attributes: a graphic figure defining its appearance when visually building a script; “isAbstract” boolean type that indicates whether an element can be used in models; “instanceMetatype” — whether an instance of this element is edge or node.

The language is limited and, in particular, it is not possible to set specific attributes for each type of actuators. For example, the volume of watering for the device watering the soil.

6. Tool Implementation

To model a scenario end-user drags all the necessary elements from the palette to the stage: actuators with sensors from his real greenhouse and operations with intervals for complex rules formation. Then he determines the values of all attributes of the selected nodes. And then he selects the edge in the palette and connects sources and receivers successively clicking on them. For the convenience of the user, the ability to draw ports on elements has been added to the editor — ports are the places on an element where edges can be connected. On the left side of the elements there are ports for incoming edges and on the right for outgoing. New free ports are automatically added to the node if all the others are already occupied but the element can still be linked to other vertices. User interface of our tool is presented in figure 6.

After the script is modeled, the user clicks on the “Generate” button on the right panel to generate code using the created model.

A generator has been implemented to produce an executable file that can be run on a micro-controller. It is written in C# and uses RX.NET and T4 technologies. T4 runtime text template takes model from repository as parameter and at the beginning creates an instance of the corresponding class for each element from this model assigning each an identifier. This identifier is used to set instance name and further to create dictionary with this identifier as a key where the “Operation” blocks can store values from all nodes connected to them. Property values of instances are set according to the attribute values from the model.

Our system should react and process the data that comes from different sensors. So each element should handle the
events that the sources of its incoming vertices sends and then generate events for subsequent elements in the rule chain. To describe this behavior of the elements and to simplify work with event flows, the RX.NET library was used. Each node from the model is considered as the Subject in terms of Reactive Extensions. This means that it is at the same time the Observer, it reacts to changes in the preceding nodes, and the Observable, it allows to subscribe to its changes. Real greenhouse sensors and actuators or robot simulator sensors are considered to be Observables and Observers respectively.

Testing of the prototype required the use of a real microcontroller. We have used the TRIK micro-controller as it can run .NET virtual machine and real sensors and actuators can be easily connected to it. The Trik-Sharp library provides access to controller devices and even directly supports Reactive Extensions, representing sensors as Observables and actuators as Observers.

An example of the code generated by the model in figure 7 is shown in the listing 1. The “Air t” node is waiting for a value from the temperature sensor. The “Interval” node subscribed to “Air t” receives this value and checks whether it lies in the (5, 15) interval. And if it does, then the “Open the window” node receives TRUE value which means that it should send the open command to the actuator on the window.

7. Experiment

To understand how easily end users can solve problems with the new language, a small test scenario was formulated. “If the temperature sensor on port #1 shows a value greater than 20 degrees — open the window with an actuator on port #5. If less — close it with an actuator #4.”

Four users were invited to model the scenario using the developed “smart greenhouse” solution. Two of them were not familiar with programming (the lawyer and the mechanical engineer). Two others were undergraduate students in software engineering familiar with UML but not with IoT. At the beginning of the experiment, work with the system was demonstrated on the example of similar scenarios. Users from the second group finished the work with the task faster, but on average it took only 3 minutes to model such scenario.

8. Lessons Learned

As mentioned earlier, “smart greenhouse” language metamodel includes only association relation and language itself is only a prototype of a full-featured programming
element0 = new Actuator(0);
element0.Port = 0;
IObservable<int> observable0 = 
    System.Reactive.Linq.Observable.FromEventPattern<int>(
        h => element0.Event += h, h => element0.Event -= h)
    .Select(e => e.EventArgs).Synchronize().DistinctUntilChanged();
IObserver<int> observer0 = Observer.Create<int>(x => element0.Action(x));
ISubject<int> reactElement0 = Subject.Create<int>(observer0, observable0);

element1 = new Interval(1);
element1.Min = null;
element1.Max = null;
IObservable<int> observable1 = 
    System.Reactive.Linq.Observable.FromEventPattern<int>(
        h => element1.Event += h, h => element1.Event -= h)
    .Select(e => e.EventArgs).Synchronize().DistinctUntilChanged();
IObserver<int> observer1 = Observer.Create<int>(x => element1.Action(x));
ISubject<int> reactElement1 = Subject.Create<int>(observer1, observable1);

element2 = new Sensor(2);
element2.Port = 0;
IObservable<int> observable2 = 
    System.Reactive.Linq.Observable.FromEventPattern<int>(
        h => element2.Event += h, h => element2.Event -= h)
    .Select(e => e.EventArgs).Synchronize().DistinctUntilChanged();
IObserver<int> observer2 = Observer.Create<int>(x => element2.Action(x));
ISubject<int> reactElement2 = Subject.Create<int>(observer2, observable2);

var sub0 = reactElement1.Subscribe(reactElement0);
var sub1 = reactElement2.Subscribe(reactElement1);

Listing 1: Example of generated code

language for greenhouses. The set of language entities is clearly defined because it affects the code generation logic, which takes into account only specific sets of possible combinations. And types of sensors and actuators that can be installed in a real greenhouse are known in advance and taken into account when creating the language. Due to the fact that the end user does not need to invent additional classes when working with a greenhouse, during the work with this language there actually was no opportunity to take advantage of the deep metamodeling for end-user scenarios. From the language designer point of view, classic two-level metamodeling architecture was used — Infrastructure Metamodel and everything below it can be considered metamodel, “smart greenhouse” language is metamodel and user scenarios of “smart greenhouse” control can be considered models. We believe that it is a typical situation in language design — to do something meaningful with models, custom code is needed, be it a generator, an interpreter, an analyzer and so on, and a need to write custom code greatly limits the flexibility of a metamodeling infrastructure.

On the other hand, deep metamodeling proved itself beneficial for DSM platform architecture. We implemented model persistence capabilities on Core Metamodel level, so each model derived from Core Metamodel can be stored/loaded. But editor capabilities are described by Infrastructure Metamodel, so we can have several different infrastructure metamodels for different visual language usage scenarios (for example, complex metamodel for an editor based on WPF and much simpler metamodel for a limited functionality editor based on Windows Forms). They are all can be based on the same Core Metamodel and take advantage of the functionality implemented in Core level. Our tool presented here did not require such capabilities, but we are planning to use them for web-based version, with completely different model editor requiring completely different infrastructure metamodel.

9. Related Work

There are some other systems that allow end users to create rules for devices sets. One of them is Node-RED — a popular and widely used open source programming tool. It uses browser for scheme of interaction between devices definition. When compared with our system, the rules built in a similar way. The difference is the wide choice of blocks on the sidebar of Node-RED and some of them provide an opportunity to set complex rules by writing JavaScript functions. There are also many libraries that allow to expand the capabilities of the tool. But such an abundance of options for designing creates a sense of a complex system for beginners and have no sense in our greenhouse case, and a need for JavaScript programming makes it unsuitable for non-programmers.

There are many DSM platforms supporting deep metamodeling and, more general, multi-level metamodeling:

Melanee [7], MetaDepth [8], WebDPF [11] to name a few. They are mainly created for research and concept-proof purposes and to our best knowledge there are no reports on their use for creation of actual domain-specific tools for end-users. Practical case studies like presented in this article are very limited. Research reported in [15] analyses features of 21 existing DSM tool with support of multi-level modeling, but lacks analysis of applicability of those features to real-world end-user applications. Work [16] considers methodological questions on evaluating multi-level modeling techniques and tools, but also is focused on technical qualities like model maintainability or model size, and does not report the experience of creating tools useful for non-programmers.

10. Conclusion

A new REAL.NET tool based on the deep metamodeling approach made it possible to quickly create a prototype of “smart greenhouse” graphical programming tool, which includes new language, editor and code generator. This prototype was tested with non-programmers and, as it seemed, turned out to be quite acceptable in real life for end users. Thus, in this article we reported a case study where a visual programming tool built using deep metamodeling architecture has enabled non-programmers to do limited programming tasks for a real-world scenarios.

Despite the fact that the area of “smart greenhouses” is rather narrow, it is part of the concept of the Internet of Things, and the result of this work can be extended to control sensors in any “smart” systems. All that is required is to determine the types of sensors that are possible to be installed in this area. And even if there are logical rules that are needed for processing values, but do not exist in the present system, the task of building a similar system based on this knowledge gained can be solved quickly using REAL.NET platform.

A logical continuation of the development of the application may be the transition to the web version. This will free the user from the obligation to install software on his own and also will help get rid of the need to create different versions for different platforms.

References


