Natural Language Processing for Intelligent Virtual Assistant System

Mykyta Syromiatnikov[1], Victoria Ruvinskaya[1]

[1] *Odessa Polytechnic State University, Shevchenko av., 1, Odesa, 65044, Ukraine*

Abstract
This article provides a result of the concept creation and development procedure of a natural language processing service for an intelligent digital assistant system. The tasks to be solved by the service have been identified, they include intent detection, named-entity recognition, question answering, ranking, multi-label classification, and text normalization. The method of deep learning model optimization based on knowledge distillation was proposed and evaluated. Also, the method for multilingual open-domain question answering using gradient boosting over decision trees, multilingual vector embeddings, and deep learning models has been developed. As a result, the NLP service, which contains data processing algorithms, an extended neural network training method aimed at inference productivity optimization, and original deep learning models showing high accuracy on test sets for English, Russian and Ukrainian languages, was developed.

Keywords
Natural language processing, machine learning, deep learning, data mining, text processing, virtual assistant system

1. Introduction

Nowadays, the growth rate of digital sphere development is increasing exponentially: a lot of modern services and applications burst into our lives and brand-new formats of interaction are becoming a reality. At the same time, outdated channels of interaction with users remain and it is almost impossible for a significant percentage of small and medium-sized businesses to fully automate their wide range of services, resulting in an excessive waste of time for people and an increase in operating expense for companies. This hypothesis was confirmed by the challenge of the COVID-19 pandemic: the rapid transition of consumers to digital channels of interaction has led to situations where the fate of companies was decided by the speed of their digitalization [1].

If you look at the super apps [2] of market leaders like Google, Facebook, Tencent, you can find one of the possible solutions that work in practice. Its essence lies in the integration of one or more natural language interfaces (NLI) [3], often the implementation is carried out in the form of a smart assistant or an intelligent helper, which is a software agent that can process user requests and independently perform tasks of various kinds with the help of the owner's data or settings. The possible functionality includes interaction with smart home devices and synchronized services such as email, calendar, music. “The term chatbot is sometimes used to refer to virtual assistants with a text chat interface.” [4]. Chatbots are usually created by using natural language processing technologies (NLP). There are also voice assistants that use speech recognition and synthesis in addition to text processing algorithms.

Due to the exponential growth of computing power and textual content in recent years, the NLP industry is developing very actively: new architectures of in-depth learning models are published, existing ones are improved, and active work is carried out to increase the training efficiency of current

Information Control Systems and Technologies (ICST-ODESSA-2021), September 23–25, 2021, Odesa, Ukraine
EMAIL: nik.syromyatnikov@gmail.com; ruvinska@opu.ua;
ORCID: 0000-0002-0610-3639 (Mykyta Syromiatnikov); 0000-0002-7243-5535 (Victoria Ruvinskaya);

state-of-the-art methods [5]. The development of the sphere is also facilitated by the expansion of possible areas of application: technologies for processing the natural language are used not only in search engines but also to automate call centers and FAQ-systems, write news [6], program code [7], etc. At the same time, for a wide range of languages, there is a problem with the quality and quantity of task-specific data. That is why a lot of research and practice is devoted to transfer learning and language model creation.

2. Description of Problem

In general, the virtual assistant system contains various functional modules that are used to generate a response and execute actions, like ordering or reservation.

Figure 1 shows one of the possible options for implementing the architecture of an intelligent assistant system, the functionality of which includes more than ten different functional modules such as search, viewing news, weather forecasts, and ordering.

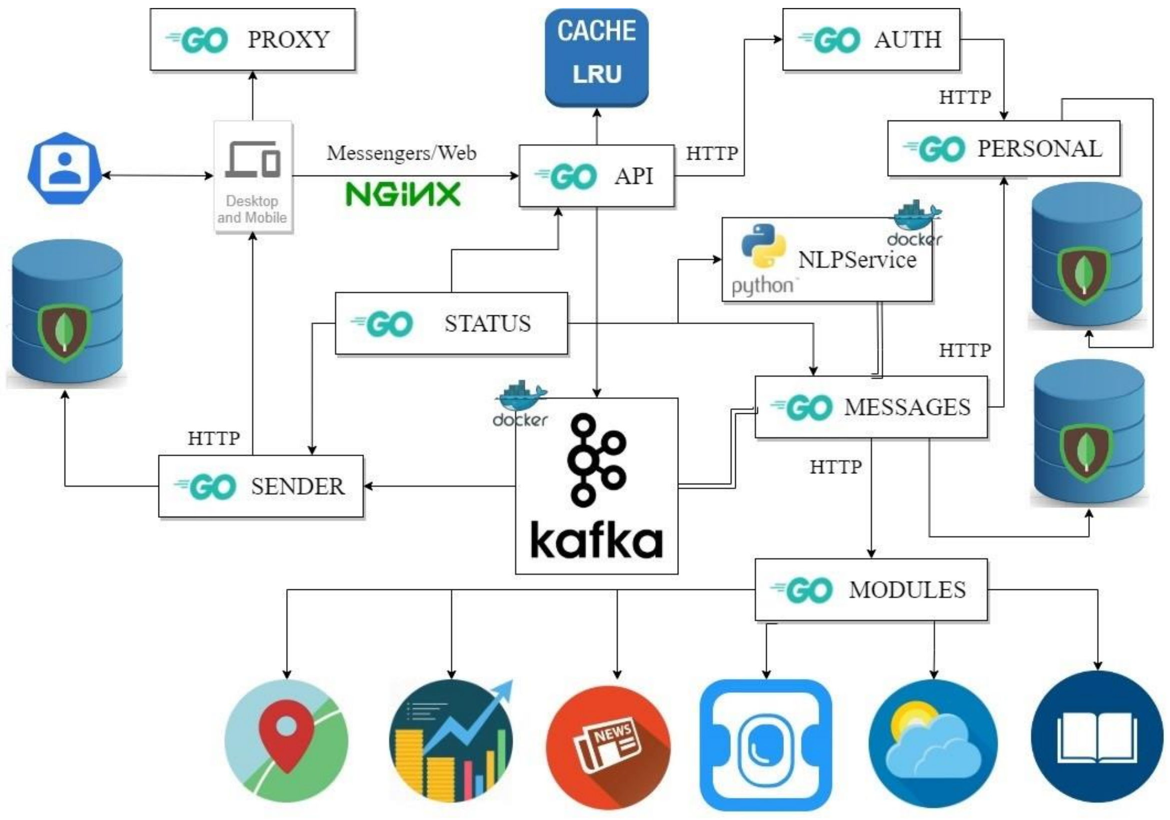


Figure 1: Virtual assistant system architecture

Microservice architecture is a quite popular design approach for this kind of product since it is aimed at decomposing functionality into separate services, minimizing the coupling [8], and maximizing cohesion [9]. It is also worth noting, that in our case, the interaction takes place using a text interface through the web application and messengers with the implemented Bot API functionality. Flowing through API and authorization, the user request enters the message processing service (MESSAGES on architectural diagram). The figure shows that MESSAGES interacts with the natural language processing service called NLPService, the development of which we are considering in this publication.

Localization for the countries of Europe and the CIS region is provided, with special attention paid to the Ukrainian, Russian and English languages. That is why it is important in our case to build an NLP service with multilingual support. Therefore, for this purpose, a combination of neural networks with a high-level context understanding, generalization ability, and machine learning algorithms with an optimal speed-accuracy trade-off would be used to develop common functionality for this domain, such as intent detection, named-entity recognition, text normalization, etc.

Most of the existing open-source solutions have a weak coverage of Eastern Europe, and high-accuracy, multifunctional systems with support for CIS languages are either difficult to integrate or extremely expensive. Therefore, the work aims to design and develop the natural language processing service for a virtual assistant system using the synthesis of standard machine learning and state-of-the-art deep learning methods with special attention paid to the productivity and accuracy of models for Russian, Ukrainian, and English languages.

3. Literature review

Various algorithms and methods of the natural language processing sphere are used for the tasks of text mining and transformation of the user's natural queries into a structural form. In narrowly focused systems, the use of approaches based on rules and regular expressions is justified [10], because they have fairly high productivity and do not require significant computational resources. However, if we are talking about open-domain virtual assistants, then their high intelligence can be achieved using deep neural networks.

One of the well-established solutions is recurrent neural networks (RNN), ideal for sequential data. Widespread RNN-type networks are LSTM and GRU, which solve vanishing/exploding gradient problems. The gated recurrent network has fewer parameters than long short-term memory but may achieve compatible accuracy in some speech recognition tasks [11].

In 2014, the Encoder-Decoder architecture with two recurrent neural networks was proposed. The Encoder encodes a sequence of characters into a fixed-length vector, and the Decoder decodes the context vector into another sequence of fixed-length characters [12]. Unlike standard recurrent models, this architecture allows obtaining the resulting sequences with a different length from the input sequence. One of the Encoder-Decoder implementations, Seq2Seq, has brought significant improvements in the quality of machine translation [13].

In recent years, the state-of-the-art results of natural language understanding and generation have been demonstrated by various implementations of the Transformer architecture published in 2017. In contrast to Encoder-Decoder, Transformer contains not recurrent and convolutional layers but dense ones along with proposed multi-head attention mechanism and positional encoding for simultaneous sequence processing.

Language models, networks that determine the probability of characters, words, n-grams occurrence in the sequence, based on Transformer architecture, such as BERT, Transformer-XL, BigBird, and autoregressive GPT and XLNet, demonstrate state-of-the-art accuracy in most tasks of the field after training on massive text corpora.

The disadvantages of this architecture include the need for significant computing resources for the training stage and inference. To address the problem of slow model computations one can use quantization to reduce weights size and degrade accuracy [14], model transfer from Pytorch/Tensorflow backend to ONNX Runtime, or knowledge distillation procedure with a smaller model creation [15].

4. Models and methods development

4.1. NLPService concept

As it was stated in the problem description section, the natural language processing service functionality includes intent detection, named-entity recognition, and text normalization. However, this is not enough for an open-domain assistant. The complete functional requirements are demonstrated in the use-case diagram (Figure 2).

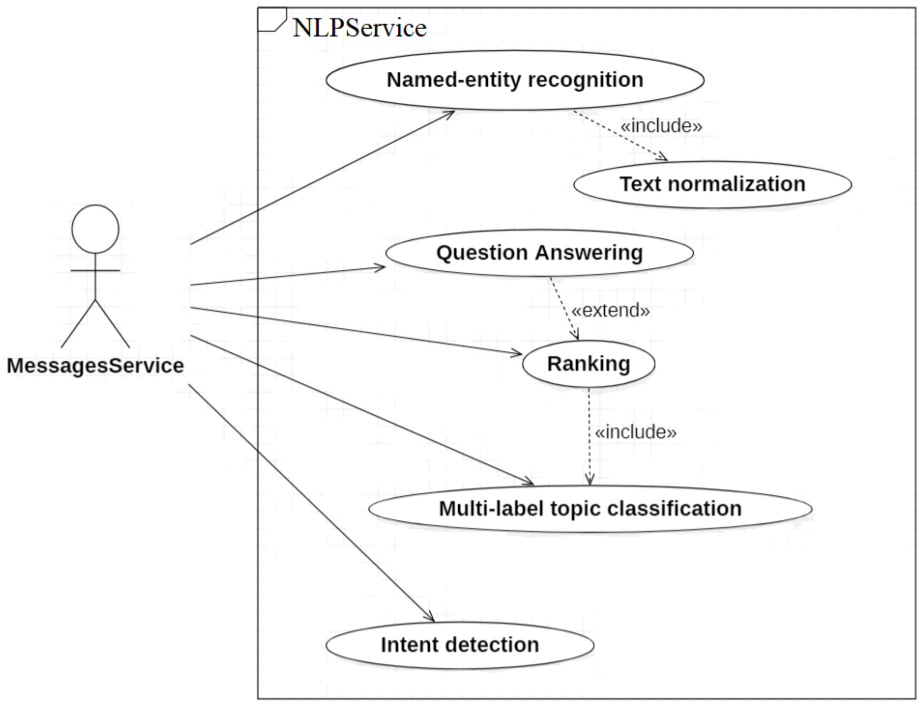


Figure 2: NLPService use cases

Generally, intent detection is the task of retrieving user intention from the input sequences of text, speech, or even images. For example, the sentence “show me the wind speed forecast for tomorrow” can be recognized as intent “weather.wind.speed”. The level of detail, in this case, depends on the specific requirements of the system, i.e., it is possible to determine the intention of "weather-forecast" instead, however, then it is necessary to introduce additional logic into the weather service to obtain a specific action after it has been called. To build the NLPService, the first option would be used, since this approach reduces the functional load on the services.

Named-entity recognition (NER) is a very important part of text mining. The NER task aims at extracting from the text sequence as many useful named entities (e.g., names, time, and date) as it is possible. These entities then flow to the service, defined by the intent, and are used to personalize answers or make relevant service responses.

Text normalization is a required stage after named entities have been recognized since in natural languages there may be various declensions of words, plurality, and genders. Most of the time, the goal of normalization is to find the normal form of the word or group of words, but in some cases, it is important to match the form with the surroundings as the context of the sequence may be changed otherwise.

Context question answering is the core part of quick answer functionality. Given the question and text paragraph, the task is to return the segment of the paragraph, which is the most relevant. For the open-domain question answering where there are multiple paragraphs, it is required to use ranking first. The purpose of this action is to find one or several relevant passages in the storage, where there can be billions of them. To speed up the ranking, multi-label topic classification, which predicts multiple labels (e.g., science, history, art) for every text sequence, would be applied.

4.2. Fine-tuning BERT for question answering task

To solve the problem of answering the question (QA), a deep learning model based on the BERT architecture (Figure 3) would be used. Bidirectional Encoder Representations from Transformers or BERT is a language model [16] based on the Transformer architecture, which is an improvement of Encoder-Decoder, where the encoder creates context vector from the input sequence and the decoder generates output based on the hidden states. A feature of the Transformer is an improved attention mechanism and the use of feedforward networks, which allows processing the entire input sequence at once, in contrast to the RNN networks. Only encoder layers are used in BERT, along with embeddings and pooling. BertEmbeddings uses tokenizer output to create a representation of the input sequence and feed it to the BertLayers. Word_embeddings layer is used to build the vectorized form of input tokens, position_embeddings introduce such an important property as the order of tokens in sequence [17]. The token_type_embeddings layer helps the model to understand where one sequence ends and another one starts, in the case of question answering those segments are question and text paragraph. For this purpose, the special token "[SEP]" is used. There is also a LayerNorm to normalize the activities of the neurons and reduce training time [18]. The dropout regularization method with a 0.1 probability to deactivate hidden units is used to reduce model overfitting.

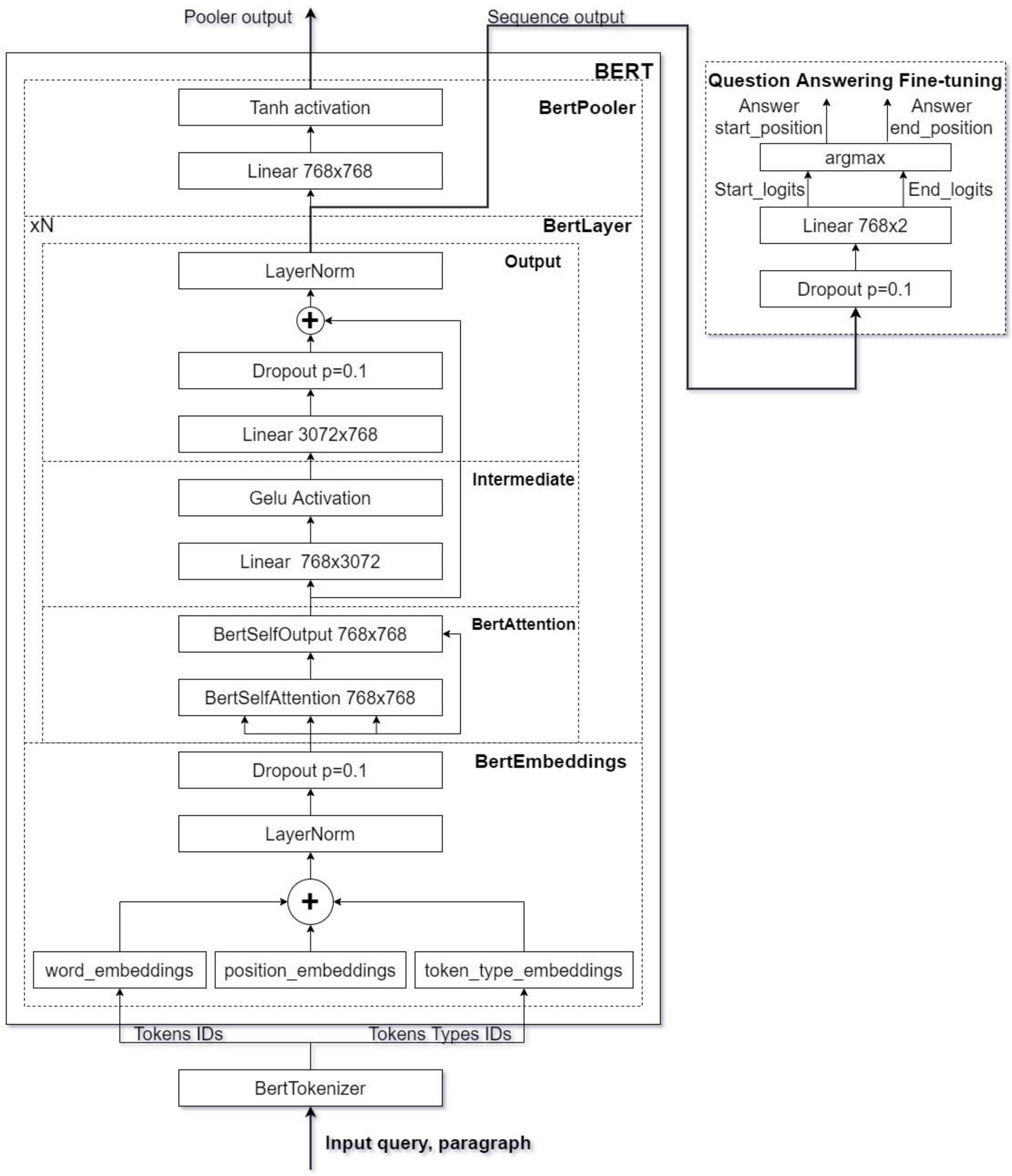


Figure 3: BERT for question answering architecture

BertLayer contains a dense layer with GELU activation and self-attention – a mechanism, which "allows the inputs to interact with each other ("self") and find out whom they should pay more attention to (“attention”)” [19]. Several successive applications of this layer with different weights allow the model to form a representation of the meaning, grammar, or other aspects of the input vector with the help of other elements of the text sequence. The size of the hidden layers and their quantity depends on the model type, for this work the 12-layer network with 768 units each was chosen.

As the BERT model pretraining includes those tasks: masked language model (MLM) [16] and next sentence prediction (NSP) [16], there are at least two possible options for output selection. The first one is the sequence output, which is the result of forward pass through all of the encoder layers sequentially for every input token. During the training, the output from hidden layers is taken for the MLM task, where the model has to predict words, which are replaced by the special mask token, in a sequence. The second one is the pooler output with a single result for the “[CLS]” token (this is the first element of input), the purpose of this type of output is to return the confidence of similarity of two text sequences for the NSP problem. However, the pooler output can also be used for various text classification tasks.

For the question answering procedure, we add fine-tuning head on top of the model. The sequence output flows to the dropout and linear layer, which returns logits for start and end positions for each input token. After that, all that remains is to apply the argmax function to get start/end token positions of the answer. Fine-tuning for 5-7 epochs on the combination of localized SQUAD v2.0 [20] and SberQuAD [21] datasets with a batch size of 20, a maximum input sequence length of 384, and scheduled learning rate with an initial value of 5e-05 shows compatible accuracy metrics.

4.3. Extended method of knowledge distillation

The inference speed of the obtained model is unaccepted for our situation. To solve this problem the extended method of knowledge distillation would be used. The method itself represents procedure, where the high-quality model (teacher) participates in the training of a smaller network (student). This significantly increases the speed of inference, as the student is not so complicated, and decreases required training time [15]. Knowledge transfer is achieved using a modified loss function:

$$L = \alpha \cdot L_{train\_loss}(\hat{y}_{student}, y_{student}) + (1-\alpha) \cdot L_{distil\_loss}(\hat{y}_{student}, \hat{y}_{teacher}) \quad (1)$$

where L is the cost function for the training procedure;
— a is the distillation parameter, which determines the influence of the teacher model;
— L_{train_loss} – loss function of student model prediction and ground truth value;
— L_{distil_loss} – loss function of student and teacher models predictions.

It can be seen from the formula that with the parameter α = 1 the standard training loss function is obtained due to zeroing of the distillation. For the training loss, one can use any task-suitable objective function, but for a distillation term in a form of the deviation of a student from the teacher, it is recommended to apply such distance functions as mean square error or a Kullback-Leibler divergence [22]. The following formula is used as the train_loss for the question answering task:

$$L_{train\_loss} = \frac{CE(start, start\_pos) + CE(end, end\_pos)}{2} \quad (2)$$

where *CE* is a cross-entropy loss function;
— *start* and *end* are logits for the start/end position of the answer. Logit refers to the tensor on which we apply the argmax function to get the position;
— *start_pos* and *end_pos* – exact indices of start/end for answer span.

At the same time, the *distil_loss* function is implemented as follows:

$$L_{distil\_loss} = t^2 \cdot \frac{KL(Ln(\frac{start}{t}), S(\frac{start_{teach}}{t})) + KL(Ln(\frac{end}{t}), S(\frac{end_{teach}}{t}))}{2} \quad (3)$$

where *KL* is a Kullback-Leibler divergence function;

— t or a temperature is a denominator used in knowledge distillation to flatten logits;
— S – softmax function, which transforms values to probability distribution;
— Ln – softmax function with the subsequent application of the natural logarithm;
— *start* and *end* are logits for the start/end position of the answer;
— $start_{teach}$ and end_{teach} are teacher model logits for the start/end position.

Our extended knowledge distillation procedure consists of the following stages:

1. Fine-tune the 12-layer BERT model for a specific task to obtain the best accuracy.
2. Build a smaller model with 4 layers - DistilBert.
3. Init student with the teacher weights from the 1st, 4th, 8th, and 12th layers.
4. Set α parameter within [0.3, 0.7] and increase the learning rate by 1-1.5 times.
5. Train student model on the extended dataset with augmented data.

The weights transfer is an important part of this method, as it helps to reduce the number of steps needed for the optimizer to find the global extrema of the objective function during training.

4.4. Fine-tuning BERT for the multi-label topic classification task

The goal of this classification is to obtain one or several labels for the input text. In our case, there are 21 topic labels: person, nature, organization, geo, location, facility, product, event, art, politics, achievement, science, news, sport, business, religion, encyclopedia, how-to, social networks, NSFW, and other. As for the QA task, the BERT model (Figure 4) would be used to solve this problem.

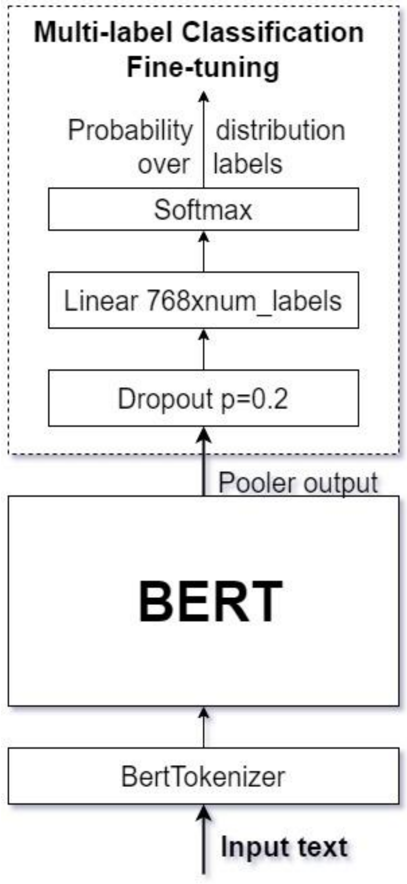


Figure 4: BERT for multi-label classification architecture

The input text sequence flows through the tokenizer to the model. After the forward pass via BERT, the pooler output is used as the fine-tuning head input. The head itself contains dropout regularization with drop probability = 0.2 and a subsequent linear layer, which has 21 outputs, one for each label. There is also a softmax layer to transform the output into a probability distribution. The last step is to pass this distribution through a threshold to zero all unmatched labels.

The multilingual dataset with 6255 records was build using Wikipedia and news articles annotation. Fine-tuning for BERT-base-multilingual-uncased took 7 epochs with binary cross-entropy as objective function, learning rate = 3e-05, sequence length = 512 and batch size = 32. A distilled 4-layers version was build based on this model. The distillation loss function is similar to those, presented in equation 1.

4.5. Fine-tuning BERT for the ranking task

Fine-tuning for the ranking task (Figure 5) is carried out using the pooler output of the BERT model, dropout with p = 0.25, linear layer, and sigmoid to transform the output into the confidence of similarity and dissimilarity.

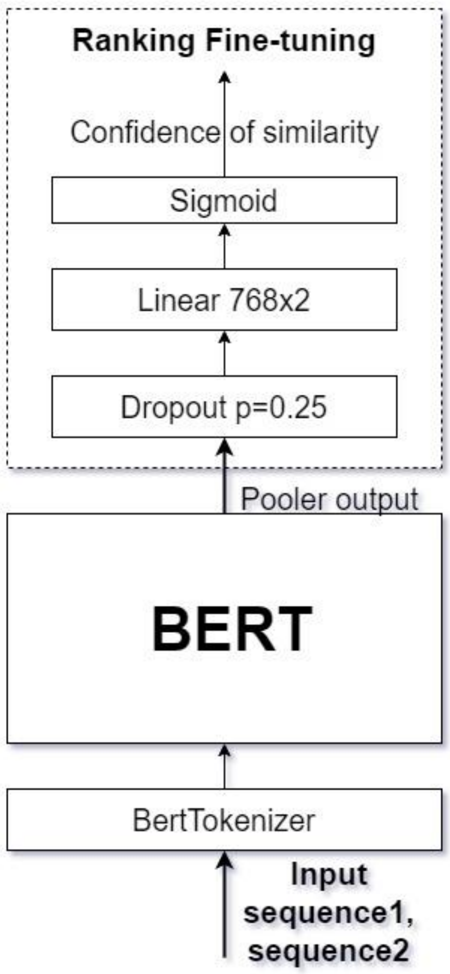


Figure 5: BERT for ranking architecture

The training dataset contains 185000 Wikipedia query-passage pairs with appropriate similarity labels in Russian, Ukrainian, and English languages. For fine-tuning procedure, those parameters were chosen: 3 epochs with learning rate = 2e-05, sequence length = 368 and batch size = 4.

4.6. Extended method of multilingual open-domain question answering

For the fast answer functionality, we use the extended method of multilingual open-domain question answering [23], the detailed architecture of which is presented in Figure 6. Here DistilBertForMultiLabels is the distilled model from the multi-label classification chapter. Laser is the technique released by Facebook AI Research to create multilingual vector embeddings [24], XGBoost – gradient boosting over decision trees model, which calculates similarity for embedding vectors of input sequences, and BertForQA is the model, needed to provide a fast answer (the most relevant fragment of text passages) on user request. The simplified algorithm is as follows:

1. Obtaining topic labels and vector embedding for the user query simultaneously.
2. Filtering stored contexts based on predicted labels.
3. Similarity calculation between the query vector and filtered contexts vectors.
4. Performing context ranking of 50 most relevant passages.
5. Searching for the most relevant segment among the top-10 passages.

As a result, this method allows to obtain a relevant text fragment, not the whole paragraph using the BertForQA model and to solve partially the problem of paraphrasing and multilingual representations with the help of DistilBertRanker.

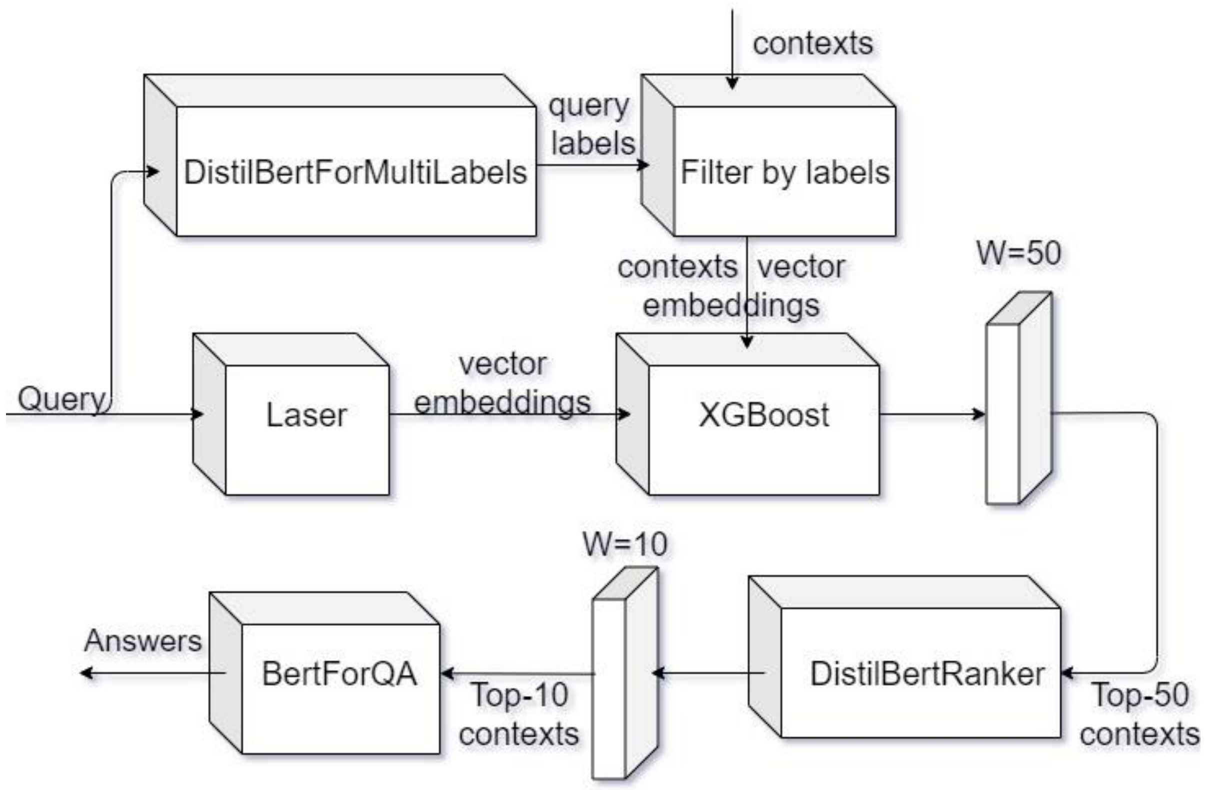


Figure 6: Multilingual open-domain question answering

4.7. Fine-tuning BERT for the intent-detection task

The problem of predicting text query intention is similar to multi-label classification: based on the input sequence we need to predict relevant tags. Except for the linear layer in the fine-tuning head, which has an output dimension of 76. The training dataset consists of multilingual queries for 76 different types of intents, such as “knowledge.search”, “products.findTickets”, “place.book”, “other.dialogue”, and other. Moreover, there are context intents (e.g. “context.fillPersonGivenName”, “context.fillGeoCity”), which would help to manage dialogue context during interaction with a user or to extract parts of complex named-entity (full name, address). The fine-tuning procedure is performed for 12 epochs with a learning rate of 3e-05, batch size = 64, and sequence length of 64.

4.8. Fine-tuning BERT for the named-entity recognition task

The NER is one of the sequence tagging tasks, where for each token of the input sequence the tag is assigned. To solve this problem, the BERT (Figure 7) trained on the dataset with 19 different types of entities from CoNLL 2003 [25] would be used. The model outputs predictions in the format of BIO-markup, where B reflects the beginning of the entity, I - the words inside the entity, and O - the absence of entity. For example, My – O, name – O, is – O, Nikita – B-Person, Syromiatnikov – I-Person.

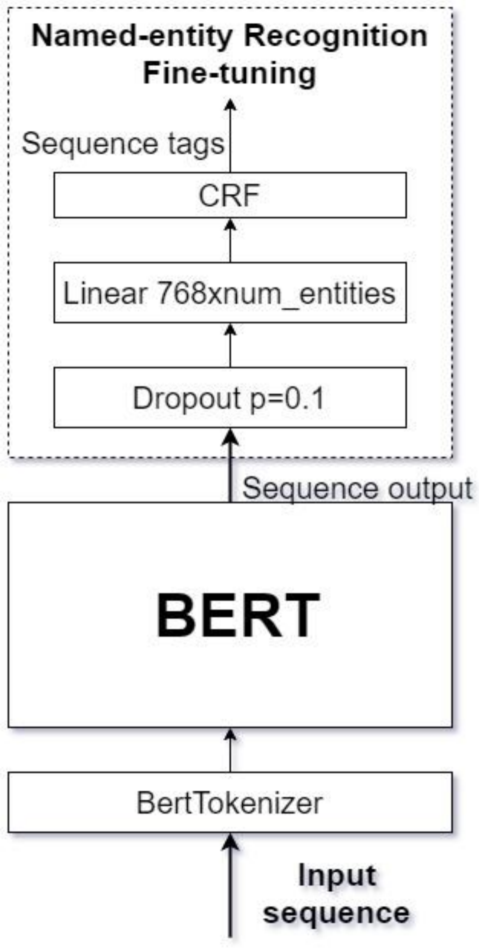


Figure 7: Bert for named-entity recognition architecture

In the figure above, the sequence output of the BERT is fed into the CRF layer through the linear layer. The CRF returns the tag sequence with the highest score.

5. Implementation

The NLPService (Figure 8) is deployed in the form of a Docker container, external requests from the MESSAGES service pass through a proxy to the corresponding endpoint of the Flask server inside. Flask handles query parameters: the called action, input text sequences, normalization parameters, and the user's settings. Then they are used to execute the appropriate action from the Model. SimBert - developed deep learning framework based on Pytorch Lightning is used to build, prototype, and run created models. For the embeddings calculation, Laser and fastText libraries are used, normalization algorithms were implemented using pymorphy2 [26] and polyglot [27].

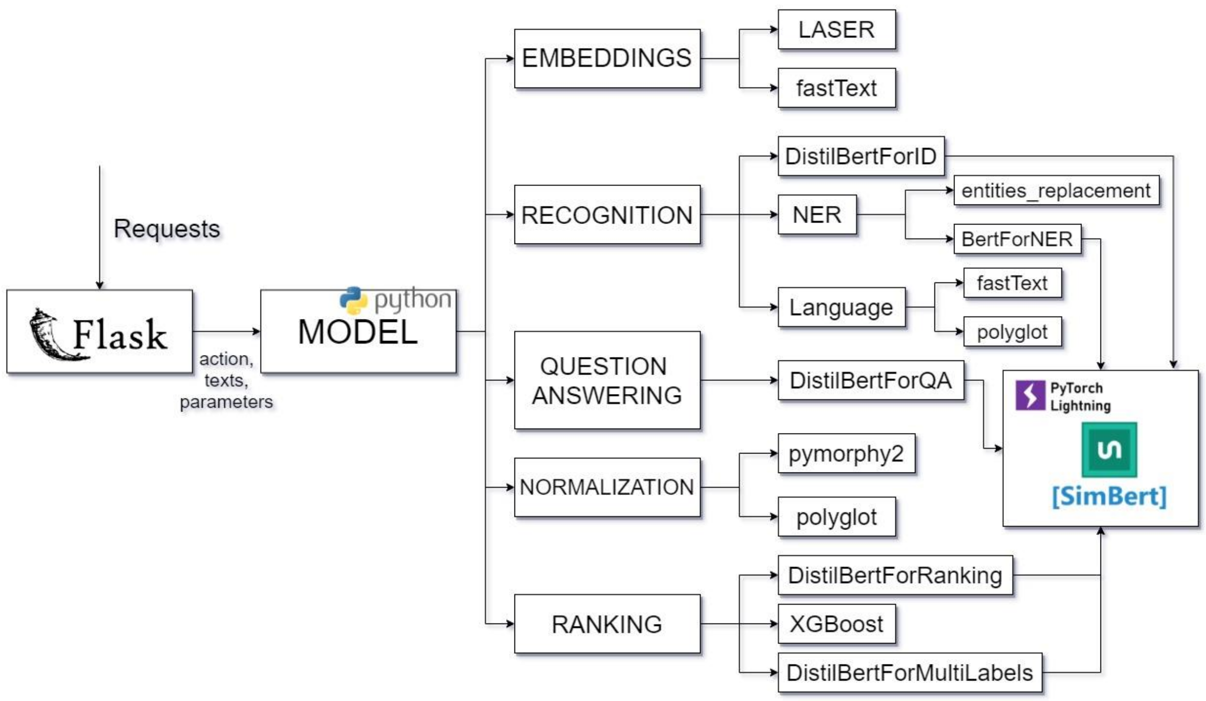


Figure 8: NLPService architecture

6. Models evaluation and NLPService testing

6.1. Models evaluation

To assess the accuracy of the predictions of the developed models on test sets, we use F1 measure - the harmonic mean of precision and recall, where precision is the fraction of true positives among true positives and false positives and recall is the fraction of true positives among true positives and false negatives.

The precision of the question answering models, however, is determined by the ratio of the number of correctly predicted words of the answer to the total predicted number of words of the answer, and recall - the ratio of correctly predicted words of the answer to the total number of words in the true answer [28]. In addition, the EM or exact match - is also used as a measure for this task. EM = 1 for the exact match of the predicted answer with the ground truth, and 0 in all other cases.

From the results (Table 1), one can see that the main metrics are the F1 for the whole test set and separately for Russian (F1 ru), Ukrainian (F1 uk), and English (F1 en) languages. Moreover, an important metric is the model performance, described by the time in seconds spent to process 100 prediction requests. It is worth noting that all of the evaluations were performed on a graphics accelerator NVIDIA RTX 2080TI. QA models testing was performed on the SQUAD v1.1 eval set to compare the results with a similar deeppavlov-squad-bert model [29]. In addition, the manually assembled test set [30] was used for the Ukrainian language. The constructed named-entities recognition model was compared with the existing deeppavlov model [31]. However, it is impractical to compare intent detection and classification models with analogs, since those tasks require a set of individual classes that meet specific requirements.

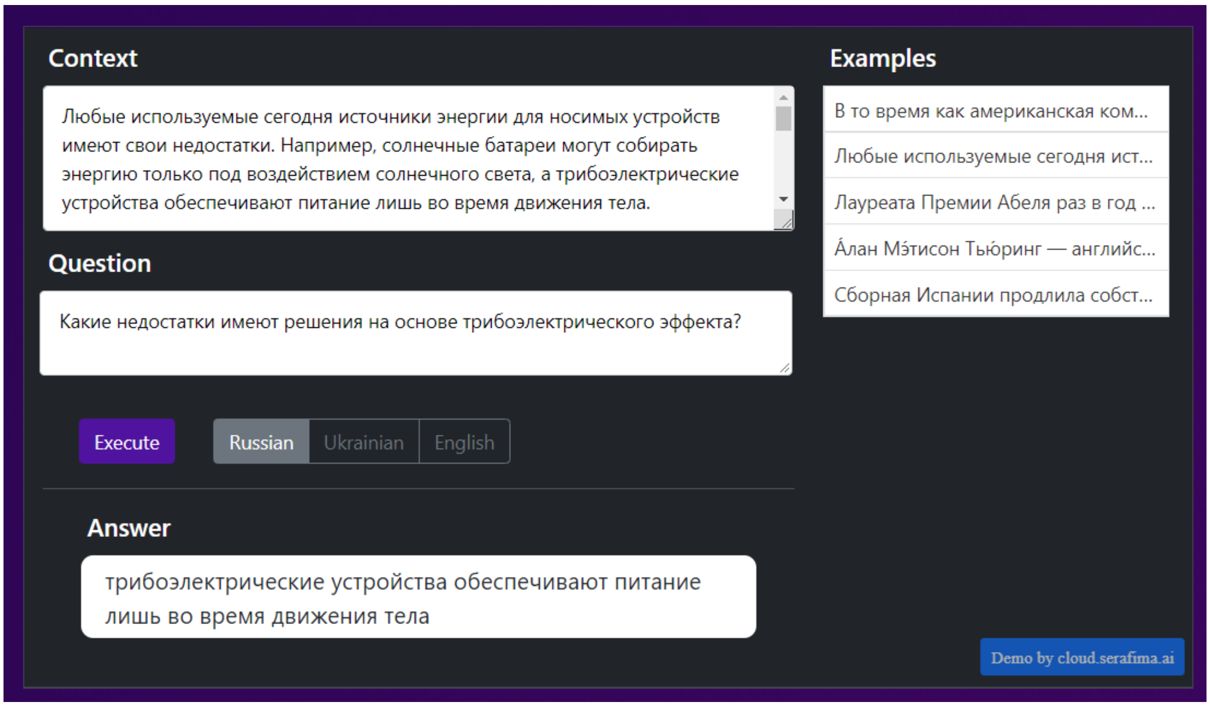
Table 1
The results of the models' evaluation

Models	F1	EM	F1 ru	F1 uk	F1 en	Exec. time for 100 requests, sec
Ranking: Laser-XGBoost	0.809	-	0.82	0.812	**0.796**	**0.015**
Ranking: BertForRanking	**0.875**	-	**0.93**	0.917	0.784	36.2
Ranking: DistilBertForRanking	0.865	-	0.928	**0.944**	0.732	12
QA: BertForQA	**0.891**	**0.813**	**0.905**	**0.921**	0.875	2.35
QA: DistilBertForQA	0.824	0.68	0.83	0.815	0.817	**1**
QA: deeppavlov-squad-bert	0.885	0.809	0.89	0.885	**0.88**	3.4
Intent detection: BertForID	**0.996**	-	**0.997**	**0.995**	**0.996**	2
Intent detection: DistilBertForID	0.993	-	0.995	0.993	0.991	**1**
NER: BertForNER	0.875	-	**0.892**	0.862	0.869	**15.2**
NER: deeppavlov-ner-mult	**0.886**	-	0.89	0.858	**0.909**	18.4

In general, the evaluation results demonstrate the ability of the developed models to compete with existing solutions due to small deviations in accuracy and a huge increase in productivity up to 3 times for distilled models.

6.2. NLPService testing

The context question answering is demonstrated in Figure 9. One can see that although the cons of triboelectric devices are not discussed directly in the text, the model correctly identified the closest paraphrased fragment of the context with the answer.


Context

Любые используемые сегодня источники энергии для носимых устройств имеют свои недостатки. Например, солнечные батареи могут собирать энергию только под воздействием солнечного света, а трибоэлектрические устройства обеспечивают питание лишь во время движения тела.

Question

Какие недостатки имеют решения на основе трибоэлектрического эффекта?

Execute | Russian | Ukrainian | English

Answer

трибоэлектрические устройства обеспечивают питание лишь во время движения тела

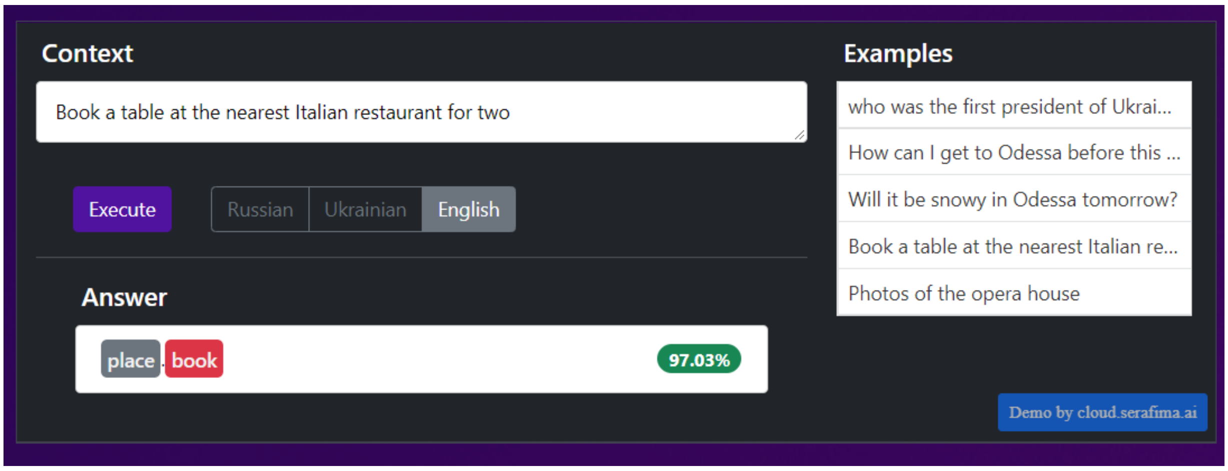
Examples

- В то время как американская ком...
- Любые используемые сегодня ист...
- Лауреата Премии Абеля раз в год ...
- Алан Мэ́тисон Тью́ринг — английс...
- Сборная Испании продлила собст...

Demo by cloud.serafima.ai

Figure 9: Question answering demonstration

Figure 10 shows the web interface for implemented intent detection functionality. For the input sequence "Book a table at the nearest Italian restaurant for two" the intent "place.book" was obtained with the confidence of 0.97, which is true, since this class is the most relevant among the available.


Context

Book a table at the nearest Italian restaurant for two

who was the first president of Ukrai...
How can I get to Odessa before this ...
Will it be snowy in Odessa tomorrow?
Book a table at the nearest Italian re...
Photos of the opera house

Figure 10: Intent detection demonstration

7. Conclusion

This material provides a concept and detailed development procedure of a natural language processing service for an intelligent digital assistant system. The tasks solved by the NLPService include intent detection, named-entity recognition, question answering, ranking, multi-label classification, and text normalization.

The method of model optimization based on knowledge distillation, which uses certain procedures for weights initialization and training hyperparameters selection, was proposed. It also simplifies the process of training multilingual four-layer models of BERT architecture for specific tasks based on the basic twelve-layer version and demonstrates 2-3 times increase of inference performance without significant deviation in accuracy for Ukrainian, Russian and English languages on the tasks of ranking, intent detection, multiclassification, and question answering.

Also, a method for open-domain question answering has been developed. It uses gradient boosting over decision trees, multilingual vector embeddings, and deep learning models, which allows obtaining a relevant text fragment and partially solves the problem of paraphrasing and multilingual representations.

8. Acknowledgments

This work was supported by the research and development program of Chernovetskyi Investment Group.

9. References

[1] A. Baig, B. Hall, P. Jenkins, E. Lamarre, B. McCarthy, The COVID-19 recovery will be digital: A plan for the first 90 days, 2020. URL: https://mckinsey.com/business-functions/mckinsey-digital/our-insights/the-covid-19-recovery-will-be-digital-a-plan-for-the-first-90-days.

[2] O. Nikolaienko, Introducing super app: a new approach to all-in-one experience, 2019. URL: https://infopulse.com/blog/introducing-super-app-a-new-approach-to-all-in-one-experience/.

[3] L. Zhou, M. Shaikh, D. Zhang, Natural Language Interface to Mobile Devices, in: Z. Shi, Q. He (Eds.), Intelligent Information Processing II. IIP 2004. IFIP International Federation for Information Processing, volume 163, Springer, Boston, MA, pp. 283-286. doi:10.1007/0-387-23152-8_37

[4] M. Hoy, Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants, Medical Reference Services Quarterly 37 (2018) 81-88. doi:10.1080/02763869.2018.1404391.

[5] J. Lee-Thorp, J. Ainslie, I. Eckstein, S. Ontanon, FNet: Mixing Tokens with Fourier Transforms (2021). arXiv:2105.03824.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, Language Models are Few-Shot Learners (2020). arXiv:2005.14165.

[7] F. Lardinois, Microsoft uses GPT-3 to let you code in natural language, 2021. URL: https://techcrunch.com/2021/05/25/microsoft-uses-gpt-3-to-let-you-code-in-natural-language.

[8] C. Larman, Applying UML and Patterns, 3rd ed., 2004.

[9] M. Fowler, Microservices Guide, 2019. URL: https://martinfowler.com/microservices/.

[10] Yargy, Rule-based facts extraction for Russian language, 2021. URL: https://github.com/natasha/yargy.

[11] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling (2014). arXiv:1412.3555.

[12] K. Cho, B. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation (2014). arXiv:1406.1078.

[13] I. Sutskever, O. Vinyals, Q. Le, Sequence to Sequence Learning with Neural Networks (2014). arXiv:1409.3215.

[14] G. Menghani, Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better (2021). arXiv:2106.08962.

[15] R. Tang, Y. Lu, L. Liu, L. Mou, O. Vechtomova, J. Lin, Distilling Task-Specific Knowledge from BERT into Simple Neural Networks (2019). arXiv:1903.12136.

[16] J. Devlin, C. Ming-Wei, L. Kenton, K. Toutanova, BERT: Pretraining of Deep Bidirectional Transformers for Language Understanding (2018). arXiv:1810.04805v2.

[17] B. Wang, L. Shang, C. Lioma, X. Jiang, H. Yang, Q. Liu, J. Simonsen, On Position Embeddings in BERT, in: International Conference on Learning Representations, ICLR 2021. URL: https://openreview.net/forum?id=onxoVA9FxMw.

[18] J. Lei Ba, J. Kiros, G. Hinton, Layer Normalization (2016). arXiv:1607.06450.

[19] R. Karim, Illustrated: Self-Attention, 2019. URL: https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a.

[20] P. Rajpurkar, R. Jia, P. Liang, Know What You Don't Know: Unanswerable Questions for SQuAD (2018). arXiv:1806.03822.

[21] P. Efimov, A. Chertok, L. Boytsov, P. Braslavski, SberQuAD – Russian Reading Comprehension Dataset: Description and Analysis (2019). arXiv:1912.09723.

[22] J. M. Joyce, Kullback-Leibler Divergence, in: M. Lovric (Eds.) International Encyclopedia of Statistical Science, Springer, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-04898-2_327.

[23] M. Syromiatnikov, O. Tsurcan, V. Ruvinskaya, Multilingual open-domain question answering model, in: Proceedings of the tenth international conference of young scientists and students, 2020, pp. 148-149.

[24] M. Artetxe, H. Schwenk, Massively Multilingual Sentence Embeddings for Zero-Shot Cross-Lingual Transfer and Beyond (2018). arXiv:1812.10464.

[25] T. Kim Sang, F. Erik, F. De Meulder, Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition, in: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL, 2003, pp. 142-147.

[26] M. Korobov, Morphological Analyzer and Generator for Russian and Ukrainian Languages, in: Analysis of Images, Social Networks, and Texts, 2015, pp 320-332.

[27] R. Al-Rfou, Polyglot, 2014. URL: https://github.com/aboSamoor/polyglot.

[28] D. Park, V. Lakshman, Question Answering on the SQuAD Dataset. URL: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2761899.pdf.

[29] Deeppavlov.ai, Question Answering Model for SQuAD dataset, 2021. URL: https://docs.deeppavlov.ai/en/master/features/models/squad.html#pretrained-models.

[30] S. Tiutiunnyk, Context-based-qa-for-uk, 2020. URL: https://git.io/JcTXD.

[31] Deeppavlov.ai, Named Entity Recognition, 2021. URL: https://docs.deeppavlov.ai/en/master/features/models/ner.html#train-and-usethe-model.