УНИВЕРСИТЕТ ИТМО

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
июль–август 2020 Том 20 № 4 ISSN 2226-1494 http://ntv.itmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
July–August 2020 Vol. 20 No 4 ISSN 2226-1494 http://ntv.itmo.ru/en/

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК УНИВЕРСИТЕТ ИТМО
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
июль–август 2020 Том 20 Номер 4

doi: 10.17586/2226-1494-2020-20-4-532-538

MODERN APPROACHES TO MULTICLASS INTENT CLASSIFICATION BASED ON PRE-TRAINED TRANSFORMERS

A.A. Solomin, Yu.A. Ivanova

Tomsk Polytechnic University, Tomsk, 634050, Russian Federation
Corresponding author: solominart@mail.ru

Article info
Received 25.05.20, accepted 28.06.20
Article in English

For citation: Solomin A.A., Ivanova Yu.A. Modern approaches to multiclass intent classification based on pre-trained transformers. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2020, vol. 20, no. 4, pp. 532–538 (in English). doi: 10.17586/2226-1494-2020-20-4-532-538

Abstract

Subject of Research. The paper considers modern approaches to the multiclass intention classification problem. The user intention is the incoming user requests when interacting with voice assistants and chatbots. The algorithm is meant for determination what class the call belongs to. Modern technologies such as transfer learning and transformers can improve significantly the multiclass classification results. **Method.** This study uses a comparative model analysis technique. In turn, each model is inlined into a common pipeline for data preparing and clearing, and the model training but with regard to its specific requirements. The following models applied in real projects have been selected for comparison: Logistic Regression + TF-IDF, Logistic Regression + FastText, LSTM + FastText, Conv1D + FastText, BERT, and XLM. The sequence of models corresponds to their historical origin, but in practice these models are used without regard to the time period of their creation but depending on the effectiveness of the problem being solved. **Main Results.** The effectiveness of the multiclass classification models on real data is studied. Comparison results of modern practical approaches are described. In particular, XLM confirms the superiority of transformers over other approaches. An assumption is made considering the reason why the transformers show such a gap. The advantages and disadvantages of modern approaches are described. **Practical Relevance.** From a practical point of view, the results of this study can be used for projects that require automatic classification of intentions, as part of a complex system (voice assistant, chatbot or other system), as well as an independent system. The pipeline designed during the study can be applied for comparison and selection of the most effective model for specific data sets, both in scientific research and production.

Keywords
natural language processing, text classification, transfer learning, transformers

Acknowledgments
The reported study was funded by the RFBR according to the research project No.18-08-00977 A. The work was partially supported by the Innovation Promotion Fund under the “UMNIK” program.

УДК 004.8 doi: 10.17586/2226-1494-2020-20-4-532-538

СОВРЕМЕННЫЕ ПОДХОДЫ К МУЛЬТИКЛАССОВОЙ КЛАССИФИКАЦИИ ИНТЕНТОВ НА ОСНОВЕ ПРЕДОБУЧЕННЫХ ТРАНСФОРМЕРОВ

А.А. Соломин, Ю.А. Иванова

Национальный исследовательский Томский политехнический Университет, Томск, 634050, Российская Федерация
Адрес для переписки: solominart@mail.ru

Информация о статье
Поступила в редакцию 25.05.20, принята к печати 28.06.20
Язык статьи — английский

Ссылка для цитирования: Соломин А.А., Иванова Ю.А. Современные подходы к мультиклассовой классификации интентов на основе предобученных трансформеров // Научно-технический вестник информационных технологий, механики и оптики. 2020. Т. 20. № 4. С. 532–538 (на англ. яз.). doi: 10.17586/2226-1494-2020-20-4-532-538

Аннотация

Предмет исследования. Рассмотрены современные подходы к решению задачи мультиклассовой классификации намерений. Под намерением пользователя понимаются входящие пользовательские запросы при взаимодействии с голосовыми помощниками и чат-ботами. Алгоритм должен определить, к какому классу относится обращение. Современные технологии, такие как трансферное обучение и трансформеры, значительно улучшают результаты мультиклассовой классификации. **Метод.** В исследовании использован метод сравнительного анализа моделей. В свою очередь, каждая модель встроена в общий конвейер для подготовки, очистки данных и обучения модели, но с учетом ее конкретных требований. Для сравнения были выбраны современные модели, которые используются в реальных проектах: логистическая регрессия + TF-IDF; логистическая регрессия + FastText; LSTM + FastText; Conv1D + FastText; BERT; XLM. Последовательность моделей соответствует их историческому происхождению, но на практике эти модели используются независимо от времени их появления, а в зависимости от эффективности решаемой проблемы. **Основные результаты.** Выполнено исследование эффективности моделей мультиклассовой классификации на реальных данных. Представлены результаты сравнения современных практических подходов. В частности, XLM подтверждает превосходство трансформеров над другими подходами. Выдвинуто предположение, по какой причине трансформеры показывают такой отрыв. Описаны преимущества и недостатки современных подходов. **Практическая значимость.** С практической точки зрения результаты этого исследования могут быть использованы для проектов, которые требуют автоматической классификации намерений, как части сложной системы (голосового помощника, чат-бота или другой системы), а также как самостоятельной системы. Пайплайн, разработанный во время исследования, можно использовать для сравнения и выбора наиболее эффективной модели для конкретных наборов данных как в научных исследованиях, так и в производстве.

Ключевые слова

обработка естественного языка, классификация текста, трансферное обучение, трансформеры

Благодарности

Исследование финансировалось РФФИ в соответствии с исследовательским проектом № 18-08-00977 А. Работа частично поддержана Фондом содействия инновациям в рамках программы «УМНИК».

Introduction

One of the recent trends in deep learning is Transfer Learning [1]. We train models to solve simple problems on a huge amount of data, and then use these pre-trained models, for solution of, more specific problems. The most well-known model is BERT (Bidirectional Encoder Representations from Transformers) [2]. This pre-trained network is a currently dominant approach for creation of models working with sequences. "The Annotated Transformer" [3] considers it in details and with examples of code on transformers and the mechanism of attention (Attention mechanism).

The General Language Understanding Evaluation (GLUE) [4] or SuperGLUE [5] benchmarks show at the top many models based on transformers (mainly BERT and its modifications).

However, two problems need to be addressed. First, whether transformers will also be effective on real data, like on a benchmark. After all, benchmarks are specially compiled datasets and disputes are ongoing in the community about the objectivity of these estimates [6]. Thus, verification of model results on real data is required and their comparison with classical approaches, such as, for example, LSTM (Long short-term memory) or CNN (Convolutional neural network) with pretrained word embeddings.

Second, not all models of transformers are multilingual, at the time of the study, respectively, and are not suitable for working with the Russian language. Therefore, two transformer models that already have multilingual versions were selected from the whole variety: BERT and XLM (Cross-lingual language model pretraining) [7].

In relation to the task of intent classification, which is a sub-task of NLU (Natural Language Understanding), we need models that not only show high theoretical results, but can be advisable to put into practice. Therefore, the logic of the study was built as follows. A dataset suitable for the task was selected, as close as possible to the real data that the algorithms may meet during the operation process. After that, the dataset was analyzed, preprocessed and visualized. A detailed description is given in "Data analysis and preprocessing" section. The next important steps were: choosing a comparison method or target metric, that is suitable for this particular task and gives the possibility to compare the quality of essentially different models (described in "Method of models comparison" section), and the models themselves for comparison (described in "Classification models" section).

The results of comparison and corresponding conclusions are given in "Results" section.

Data analysis and preprocessing

The open source dataset from the NGHack 2019 competitions, which took place in December 2019, was used. It contains 61581 users requests to one of the mobile operators. The data format is a text in Russian language. The dataset structure:

— Id – unique identifier;

— Text – text of the users request;

— Label – one of fourteen target classes(topics).

The distribution of the data between fourteen target classes is shown in Fig. 1.

The diagram shows that the classes are unbalanced. The most popular is a zero class — «мобильная связь — тарифы». Also, as a result of basic statistics calculation, it turned out that the most popular token is «тариф» and there are three words in each request on the average.

Text preprocessing was performed using standard methods of morphological and syntactic preprocessing.

- Tokenization — splitting sentences to words.
- Normalization — all words were lowercased, punctuation marks were removed, the abbreviations were spelled out. In addition, the stop words were removed. Source data was polluted with insignificant

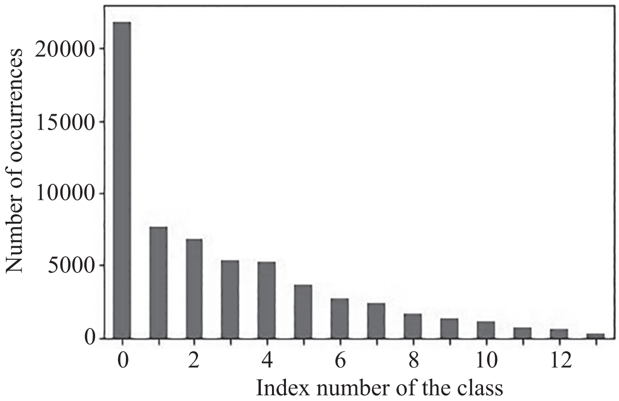


Fig. 1. Target classes distribution

words and curses, words in the other languages written in transliteration. For that purpose, the standard stop word dictionary has been expanded.

— Lemmatization — After lemmatization, all words were presented in their canonical form (infinitive for a verb, nominative singular — for nouns and adjectives).

Natural Language Toolkit (NLTK) and pymorphy2 (morphological analyzer for Russian language) libraries were used for preprocessing. The result of preprocessing is shown in Table 1. In the basic form of words the intentions can be defined clearly.

Due to the fact that different models require different preprocessing methods, transformers, in particular, use their own tokenizers. The above-mentioned ones describe the general approach to preprocessing carried out in the course of scientific research. When applying specific models, this approach was adjusted as necessary.

Method of models comparison

An important question is how to compare such essentially different models. The first and most intuitive metric is the *accuracy metric*. In this case, it is not applicable because the target class is unbalanced. Accordingly, we need metrics to assess the quality of the algorithm on each of the classes separately, for example, precision and recall metrics.

Table 1. Ten most frequent tokens before and after preprocessing

Before preprocessing	After preprocessing
как	тариф
тариф	проверить
номер	узнать
на	подключить
для	интернет
меня	отключить
тарифы	номер
можно	сколько
узнать	баланс
интернет	услуга

Precision can be interpreted as the fraction of objects that are called positive by the classifier and are positive at the same time, and recall shows what proportion of objects of a positive class from all objects of a positive class the algorithm has found.

There are several different ways to combine precision and recall into an aggregate quality criterion. *F*-measure is a harmonic mean precision and recall.

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2\cdot\frac{\text{precision}\cdot\text{recall}}{\text{precision} + \text{recall}}.$$

The *F*-measure reaches its maximum with completeness and accuracy equal to unity, and is close to zero if one of the arguments is close to zero. For this task, it was decided to use this particular metric.

To improve the generalization ability of the algorithms, it was decided to use cross-validation. The essence of any type of cross-validation is emulation of a data set that is not involved in training, but for which the correct answers are known. In our case, when the classes are unbalanced, the improved *k*-fold cross-validation method can be used called stratified *k*-fold. As in the usual method, the training sample is divided into k disjoint equal in volume parts. Then $k \times k$ iterations are performed. The following happens at each iteration (Fig. 2):

— the model is trained on the $k - 1$ part of the training sample;

— the model is tested on the part of the training sample that did not participate in the training;

— each of the *k* parts is used once for testing. Generally, $k = 10$ (5 in case of small sample size).

But in the case of stratified *k*-fold, the splitting occurs in such a way that each fold contains approximately the same percentage of samples of each target class as the complete set, that is, the whole distribution does not change. Such validation is just correct for the multi-class classification problem.

Classification models

TF-IDF + Logistic Regression. Before using complex machine learning approach, we need to check how more simple models will cope with the task. One of the approaches is to use a TF-IDF (TF — Term Frequency, IDF — Inverse Document Frequency) vector. The TF-IDF-

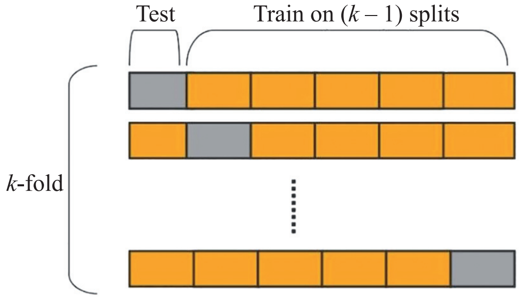


Fig. 2. Illustration of basic *k*-fold principles

vector takes into account the occurrence of each word in the user's phrase and the total occurrence of words in the collection. Words that are often found in different texts have less weight in this vector representation.

The model with TF-IDF and the logistic regression classifier was chosen as a baseline. This combination gives a good quality of classification. The completeness is much higher than when using the dictionary, with comparable accuracy: no need to invent which words correspond to which intent.

But this model also has disadvantages:

- limited vocabulary; you can get weight only for those words that are in the training sample;
- rephrasing is not taken into account;
- the order in which the words occur in the text is not taken into account.

Rephrasing is generally a separate issue. TF-IDF vectors can only be close for texts that intersect in words. The proximity between the vectors can be calculated through the cosine of the angle between them. And there are situations when it is obvious that this is the same intent and the same class, but there is no such dependence in vector representation.

The following parameters were selected for the regression:

LogisticRegression (random_state = self.seed, n_jobs = = 5, solver = 'LBFGS', multi_class = 'multinomial')

LBFGS is an optimization algorithm in the family of quasi-Newtonian methods that approximates the Bruden-Fletcher-Goldfarb-Channo (BFGS) algorithm using a limited amount of computer memory. This is a popular machine learning parameter estimation algorithm. The task of the algorithm is to minimize over unlimited values of the real vector, where it is a differentiable scalar function.

Logistic Regression with FastText. Here we are returning to the question, what to do with paraphrasing. For example, instead of a number, you can represent a word as a whole vector, as we did with TF-IDF — this is called "word embedding". One of the most popular models for solving this problem is called word2vec. One of the ways of word2vec learning works as follows: a text is an input to the model, a word is randomly selected from the context and excluded, then the next random word is taken from the context and both words are represented as hot vectors. A hot vector is a vector in accordance with the dimension of the dictionary, where only the coordinate corresponding to the index of the word in the dictionary has the value "one", the rest — "zero".

It would seem that all is well, except that some words in your matrix may not be, because the model did not see them during training. In order to deal with the problem of unfamiliar words (out-of-vocabulary), they came up with a modification of word2vec — FastText.

FastText works as follows: if the word is not in the dictionary, then it is divided into symbolic n-grams; for each n-gram embedding is taken from the matrix of embeddings of n-grams (which are trained like word2vec); the embeddings are averaged, and a vector is obtained [8].

However, the disadvantages remain. The model is not used for the entire text vector. To get a common text vector, you need to think of something: average, or average with multiplication by IDF-weights, and this can work differently in various tasks.

The vector for one word is still one, regardless of the context. Word2vec trains one word vector for any context in which the word occurs. For the task, the Embeddings for the Russian language trained on Twitter were used and the same Logistic regression as a classifier.

CNN(Conv1D) with FastText. This classification model is the same FastText, but with a neural network classifier and is used to move more complex models. In particular, this architecture is with a 1D-convolution layer (Fig. 3).

On the Embedding layer, we are embedding the matrix obtained using the get_word_vector method from FastText. This layer is not trainable, it is frozen. In short, embedding is a mapping of a point in some multidimensional space to an object, in our case, a trigram. So, we take embedding for each trigram in our text and just put all the vectors in a row, getting the desired matrix [9].

After the matrix is figured out, the issue about the convolution is considered. It turns out that the convolution can be performed only along one axis — in width. Therefore, in order to distinguish from standard convolution, it is called one-dimensional (1D convolution). The number of filters in it is 100, and the width of the window for filters is 5, activation funcion is ReLU, window step size(stride) is 1.

After ReLU the GlobalMaxPool1D layer comes. "Global" in this case means that it is taken along the entire length of the incoming sequence. Imagine that in the convolutional layer the filter matrix is fixed and is unit (that is, multiplying by it does not affect the input data in any way). And instead of summing up all the multiplication results (input data according to our condition), we simply select the maximum element. This is max-pooling. We use the Adam optimizer, loss = 'binary_crossentropy', activation 'sigmoid'. This architecture is represented by Fig. 4.

LSTM with FastText. The key to LSTM is the cell state. A cellular state is something like a conveyor belt. It moves right along the entire chain with only small linear interactions. Information can simply flow through it unchanged.

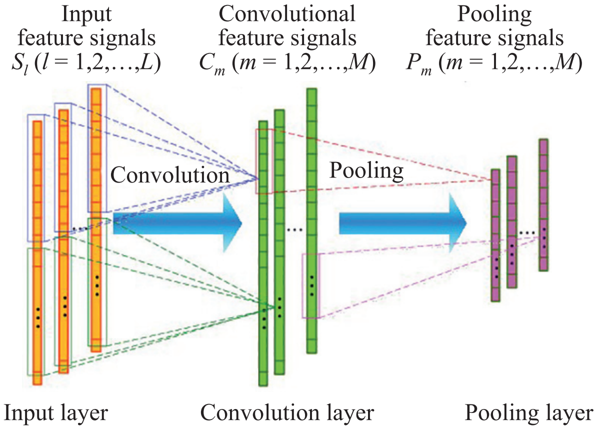
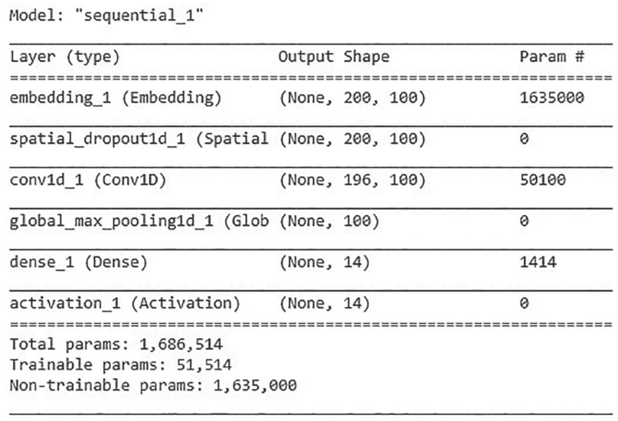

Input feature signals S_l (l = 1,2,…,L)
Convolutional feature signals C_m (m = 1,2,…,M)
Pooling feature signals P_m (m = 1,2,…,M)
Convolution
Pooling
Input layer
Convolution layer
Pooling layer

Fig. 3. CNN basic architecture for NLP tasks



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 200, 100)          1635000
_________________________________________________________________
spatial_dropout1d_1 (Spatial (None, 200, 100)          0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 196, 100)          50100
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 14)                1414
_________________________________________________________________
activation_1 (Activation)    (None, 14)                0
=================================================================
Total params: 1,686,514
Trainable params: 51,514
Non-trainable params: 1,635,000
_________________________________________________________________
```

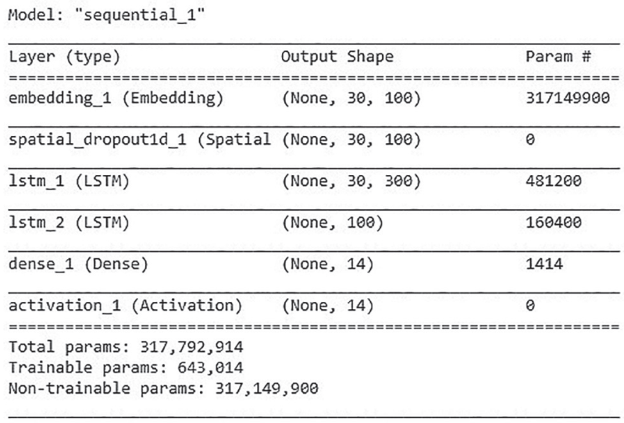
Fig. 4. CNN architecture for multiclass classification task

LSTM has the ability to remove or add information to the cellular state, but this ability is carefully regulated by structures called gates.

Gates are a way to selectively pass information. They are composed of a sigmoid layer NS and the operation of pointwise multiplication.

The sigmoid layer outputs a number between zero and one, thus describing how much each component should be passed through the valve. Zero – "skip nothing", one – "skip everything".

1. The gate of residual memory (remember gate), is also called the gate of forgetting (forget gate). We want the model in the learning process to form a special mechanism of forgetting: when new input information arrives, the model must know which knowledge should continue to be remembered and which should be forgotten.
2. The save gate is also called the input gate. When a model sees new information, it should determine whether to add it to long-term memory or not.
3. The focus gate, is also called the attention gate, or output gate. Finally, the model should determine which elements of long-term memory can be useful in the very near future. Therefore, instead of application of all long-term memory, the network learns to focus on its certain elements [10]. This architecture is represented by Fig. 5.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 30, 100)           317149900
_________________________________________________________________
spatial_dropout1d_1 (Spatial (None, 30, 100)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 30, 300)           481200
_________________________________________________________________
lstm_2 (LSTM)                (None, 100)               160400
_________________________________________________________________
dense_1 (Dense)              (None, 14)                1414
_________________________________________________________________
activation_1 (Activation)    (None, 14)                0
=================================================================
Total params: 317,792,914
Trainable params: 643,014
Non-trainable params: 317,149,900
_________________________________________________________________
```

Fig. 5. LSTM architecture for multicalss classification task

BERT. Unlike previous models of language representations, BERT is intended for preliminary preparation of deep bidirectional representations from unlabeled text by the joint preparation of both left and right contexts at all levels. As a result, the pre-trained BERT model can be finely tuned with only one additional output layer to create modern models for a wide range of tasks, such as answering a question and outputting a language, without significant tasks and specific architecture modifications.

BERT is an auto-encoder (autoencoder, AE). It hides and spoils some words in the sequence and tries to restore the original sequence of words from the context. This leads to the model disadvantages: each hidden word is predicted individually. We lose information about the possible connections between the masked words. The paper gives an example called "New York". If we try to predict independently these words in the context, we will not take into account the relationship between them.

There is a mismatch between the phases of training the BERT model and the use of the pre-trained BERT model. When we train the model - we have hidden words ([MASK] tokens), when we use the pre-trained model, we do not already supply such tokens to the input.

And yet, despite these problems, BERT showed state-of-the-art results on many natural language processing tasks [11].

For BERT model, Adam optimizer was used, with an initial learning rate of 10^{-5}, batch size of 16 and 3 epochs of training. The validation and test accuracies were taken for all experiment setups, performing each setup five times for stratified k-fold cross validation with $k = 5$. Training was on single NVIDIA Tesla P100.

XLM. XLM uses the well-known preprocessing technique (BPE) and a bilingual learning engine with BERT to study the relationship between words in different languages. The model is superior to other models in the multilingual classification problem (offering sentences in 15 languages) and improves significantly machine translation when a pre-prepared model is used to initialize the translation model.

Cross-lingual BERT for Hard BERT classification was trained in more than 100 languages; it was not optimized for multilingual models because most of the vocabulary is not distributed between languages, and, therefore, general knowledge is limited. To overcome this fact, XLM modifies BERT as follows.

First, instead of using a word or characters as input to the model, it uses Byte Pair Coding (BPE), which splits the input into the most common subwords in all languages, thereby increasing the overall vocabulary between languages. This is a general preprocessing algorithm, and a summary of it can be found here.

Secondly, it updates the BERT architecture in two ways.

1. Each training sample consists of the same text in two languages, while in BERT each sample is built in one language. As in BERT, the goal of the model is to predict masked tokens, however, in the new architecture, the model can use the context of one language to predict tokens in another, and since different words are masked words in each language (they are chosen randomly).

2. The model also receives the language ID (unique identifier) and the order of tokens in each language, that is, positional coding, separately. New metadata helps the model learn the relationship between related tokens in different languages [2].

Setups for a model were created the same as for BERT to facilitate analysis and comparison.

Results

The results of the study are shown in the Table 2. The results of the study are shown in Table 2. As we see, transformers show a significant gap in *F*-score compared to the baseline model. Neural network architectures with no attention mechanism — CNN and LSTM are close to them. There is a clear line of progress on real data in the development of approaches to the multiclass classification problem.

The architecture peculiarity of transformers takes into account the entire context at once, unlike the other models, where the context length is much smaller (hundreds of words, against dozens). Due to the concurrent processing of all tokens in the attention module, the model needs more information about the position of each token. By adding a fixed value to each token based on its position (e.g. sinusoidal function) — a step named Positional Encoding — the network can successfully learn relations between tokens. Supposedly, this is what enables transformers to show such results. If it is possible to come up with some alternative approaches with the same essence based on, for example, LSTM or CNN, then they will theoretically show results comparable to the results of transformers.

Table 2. Results of the experiments

Model	Result (*F*-measure)
TF-IDF + Logistic Regression	0.72
Logreg + FastText	0.55
LSTM + FastText	0.77
Conv1D + FastText	0.83
BERT	0.91
XLM	0.94

Conclusion

As a result of the work, the following models were studied and implemented as part of solving the intent-classification problem: Logistic Regression + TF-IDF, Logistic Regression + FastText, LSTM + FastText, Conv1D + FastText, BERT, XLM. The transformers models showed the best results with *F*-measure equal to 0.91 for BERT and 0.94 for XLM. In particular, XLM confirms the superiority of transformers over other approaches.

However, the use of these models in business solutions is still difficult for a number of reasons. Classical approaches, although not of high quality, are easy for implementation and interpretation. Transformers do not have such important properties. However, as we have seen, modern technologies, such as transfer learning and transformers, can improve significantly the results for the multiclass classification. Consequently, we can conclude that in the near future these approaches will become classic for using in chatbots and voice assistants, and they will be replaced by even more advanced models. The fact is, that during the realization of this study, about ten new transformer architectures appeared, which also require research and performance evaluations.

References

1. Ruder S., Peters M.E., Swayamdipta S., Wolf T. Transfer learning in natural language processing. *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2019)*, 2019, pp. 15–18. doi: 10.18653/v1/N19-5004
2. Devlin J., Chang M.-W., Lee K., Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2019)*, 2019, pp. 4171–4186. doi: 10.18653/v1/N19-1423
3. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser Ł., Polosukhin I. Attention is all you need. *Proc. 31st Annual Conference on Neural Information Processing Systems (NIPS 2017)*, 2017, pp. 5999–6009.
4. Wang A., Singh A., Michael J., Hill F., Levy O., Bowman S.R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *Proc. 7th International Conference on Learning Representations (ICLR 2019)*, 2019.
5. Wang A., Pruksachatkun Y., Nangia N., Singh A., Michael J., Hil F., Levy, O., Bowman S.R. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *Advances in Neural Information Processing Systems*, 2019, vol. 32.
6. Chollet F. *On the measure of intelligence*. Available at: https://arxiv.org/pdf/1911.01547.pdf (accessed: 20.04.20)
7. Conneau A., Lample G. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems*, 2019, vol. 32.
8. Mikolov T., Sutskever I., Chen K., Corrado G., Dean J. Distributed representations of words and phrases and their compositionality. *arXiv:1310.4546*, 2013.

Литература

1. Ruder S., Peters M.E., Swayamdipta S., Wolf T. Transfer learning in natural language processing // Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2019). 2019. P. 15–18. doi: 10.18653/v1/N19-5004
2. Devlin J., Chang M.-W., Lee K., Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding // Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2019). 2019. P. 4171–4186. doi: 10.18653/v1/N19-1423
3. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser Ł., Polosukhin I. Attention is all you need // Proc. 31st Annual Conference on Neural Information Processing Systems (NIPS 2017). 2017. P. 5999–6009.
4. Wang A., Singh A., Michael J., Hill F., Levy O., Bowman S.R. GLUE: A multi-task benchmark and analysis platform for natural language understanding // Proc. 7th International Conference on Learning Representations (ICLR 2019). 2019.
5. Wang A., Pruksachatkun Y., Nangia N., Singh A., Michael J., Hil F., Levy O., Bowman S.R. SuperGLUE: A stickier benchmark for general-purpose language understanding systems // Advances in Neural Information Processing Systems. 2019. V. 32.
6. Chollet F. On the measure of intelligence [Электронный ресурс]. URL: https://arxiv.org/pdf/1911.01547.pdf (дата обращения: 20.04.20)
7. Conneau A., Lample G. Cross-lingual language model pretraining // Advances in Neural Information Processing Systems. 2019. V. 32.

9. Kim Y. Convolutional neural networks for sentence classification. *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014, pp.1746–1751. doi: 10.3115/v1/D14-1181
10. Mikolov T., Karafiát M., Burget L., Cernocky J., Khudanpur S. Recurrent neural network based language model. *Proc. 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, 2010, pp. 1045–1048.
11. Vasilev I. *Advanced Deep Learning with Python: Design and implement advanced next-generation AI solutions using TensorFlow and PyTorch*. Packt Publishing Ltd, 2019, pp. 260–264.

8. Mikolov T., Sutskever I., Chen K., Corrado G., Dean J. Distributed representations of words and phrases and their compositionality // arXiv:1310.4546. 2013.
9. Kim Y. Convolutional neural networks for sentence classification // Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2014). 2014. P. 1746–1751. doi: 10.3115/v1/D14-1181
10. Mikolov T., Karafiát M., Burget L., Cernocky J., Khudanpur S. Recurrent neural network based language model // Proc. 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010). 2010. P. 1045–1048.
11. Vasilev I. Advanced Deep Learning with Python: Design and implement advanced next-generation AI solutions using TensorFlow and PyTorch. Packt Publishing Ltd, 2019. P. 260–264.

Authors

Artem A. Solomin — Postgraduate, Tomsk Polytechnic University, Tomsk, 634050, Russian Federation, ORCID ID: 0000-0003-3853-8802, solominart@mail.ru

Yuliya A. Ivanova (Bolotova) — PhD, Associate Professor, Tomsk Polytechnic University, Tomsk, 634050, Russian Federation, Scopus ID: 56641841600, ORCID ID: 0000-0002-8161-3575, Julya.bolotova@gmail.com

Авторы

Соломин Артем Алексеевич — аспирант, Национальный исследовательский Томский политехнический Университет, Томск, 634050, Российская Федерация, ORCID ID: 0000-0003-3853-8802, solominart@mail.ru

Иванова (Болотова) Юлия Александровна — кандидат технических наук, доцент, Национальный исследовательский Томский политехнический университет, Томск, 634050, Российская Федерация, Scopus ID: 56641841600, ORCID ID: 0000-0002-8161-3575, Julya.bolotova@gmail.com