УДК 519.21 https://doi.org/10.17721/1812-5409.2023/2.3

В. В. Голомозий[1], *к.ф.-м.н., доц.*
Ю. С. Мішура[1], *д.ф.-м.н., проф.*
Т. О. Яневич[1], *к.ф.-м.н., доц.*
І. О. Ізарова[1], *д.ю.н., проф.*

Порівняння 2D згорток та щільних нейронних мереж в моделях природної обробки мови з кількома вхідними реченнями

[1] Київський національний університет імені Тараса Шевченка, 01601, Київ, вул. Володимирська, 64/13, Україна

V. V. Golomoziy [1], *PhD, Prof. Associate*
Yu. S. Mishura[1], *Dr. Sci., Professor*
T. O. Ianevych[1], *PhD, Prof. Associate*
I. O. Izarova[1], *Dr. Sci., Professor*

Comparison of 2D convolutions and dense neural networks for natural language processing models with multi-sentence input

[1]Taras Shevchenko National University of Kyiv, 64/13, Volodymyrska Str., Kyiv, 01601, Ukraine,

e-mails: vitaliy.golomoziy@knu.ua, yuliyamishura@knu.ua, tetianayanevych@knu.ua, irina.izarova@knu.ua

Статтю присвячено аналізу судових справ, який грунтується на текстовій інформації, що включає позовні вимоги, підставу позову та відзив відповідача. Беручи до уваги ці параметри, ми можемо класифікувати справу до однієї із семи категорій, визначених для цілей цього дослідження, та спрогнозувати рішення по ній у суді першої інстанції. Для розв'язання цієї задачі ми використали нейронну мережу на основі представлення текстових даних згенерованих мовленнєвою моделлю XLM\RoBERTa. Було порівняно два підходи до побудови такої нейронної мережі. Один базується на вертикальному об'єднанні числових представлень декількох речень таким чином, що вони утворюють матрицю, а потім застосуванні до неї 2D згорток. Другий підхід полягає в послідовному об'єднанні речень та застосуванні щільних нейронних мереж. Останній підхід показує трохи кращі результати в нашому експерименті, в той час як перший демострує простіший процес навчання.

Ключові слова: обробка природної мови, мовленеві моделі, згортки, щільні нейронні мережі

This paper is devoted to the analysis of court cases based on multiple sentences that represent plaintiff's claim, claim motivation and defendant's response. Based on these parameters we classify a given case into one of seven categories designed for our task and then predict its decision in the first court's instance. We use fine-tuned XLM\RoBERTa for this task. There were compared two approaches for building fine-tuned model's head. One is based on stacking the numerical representation of multiple sentences so that they form a matrix and applying 2D convolutions. Second approach is based on concatenated sentences and application of dense neural networks. The latter demonstrates a slightly better performance in our experiments, while the former exhibits the simpler training process.

Key Words: Natural Language Processing , language models, convolutions, dense neural networks

Communicated by Prof. M. P. Moklyachuk

1 Introduction

Our research group is working on the project whose main aim is to find the best way of using AI for analyzing court decisions of the Unified state register of Ukraine (more then 100 million cases) to gain efficiency, speed, and valuable insights. This enables us to make faster and more accurate legal research and decision-making, and provide rational recommendations for the developing of the judicial system.

The analysis of legal documents, such as contracts and claims, constitutes a specific field of research and application of Natural Language Processing (NLP) algorithms. Special considerations in such analysis include: length of a document, specific legal language, special requirements for the legal accuracy of the generated texts, etc. The

length of a document or a series of documents typically exceeds a limit of 512-2048 tokens per sequence required by most open source Large Language Models (LLM), such as BERT (see [1]). In this paper, we construct an algorithm to predict court decisions based on documents related to civil procedure. The cases may have tens of associated documents, with several thousands of tokens each. For more details about the dataset see [2].

Another important aspect of the algorithm under consideration is the multi-sentence input. Typical NLP algorithms follow the standard transformer approach to input. This approach supposes three stages of input pre-processing. In the first stage, every input sentence is trimmed or padded so that all sentences have the same length. In the second stage, every word is encoded using some standard word embedding, thus creating a vector representation of each word. Such vectors then stack one on top of another, forming an input matrix. Finally, this matrix is added to the matrix of the same size, representing so-called positional encodings. The result is a two-dimensional matrix that represents a single input sentence.

In our project we are working on prediction of various characteristics of the civil procedure. The characteristics of interest are: first instance decision, whether it was appealed, decision in appeal, final decision, the process duration, associated costs and etc. Predicting these values based on the whole text of the case documents is a nearly unfeasible task since civil procedure may last for years, and a stock of associated documents may include tens or even hundreds of various documents. To circumvent this problem, we have manually labelled a subset of documents specifying particular sentences or text fragments that, we think, are relevant to the algorithm we develop. Among others, we have chosen claim, claim motivation and defendant's response as three main elements to base our algorithm on. Each of these elements is a relatively short fragment of text that can easily be passed through the existing open-source LLMs. But each LLM takes only one sentence as input, while we have three. Our approach is to use LLMs to derive a contextual numeric representation of each text fragment and then build another neural network to predict values of interests based on these representations. As mentioned above, we use claim, claim motivation and response as input text fragments, and after passing them through the LLM, we have got three 768-dimensional vectors.

When stacking these vectors, we got a 3×768 matrix as input to our algorithm. Since matrix-type inputs are typical for convolutional neural networks widely used in computer vision, we decided to utilize a convolution-based approach. It is worth mentioning that the use of convolutions in text analysis is not an entirely new idea. See, for example, [10]. As an alternative approach, we can use a dense neural network with all our inputs being flattened into one long vector.

We developed and evaluated three models (small, medium and large) for each approach and compared the results. As a result of our experiments, we have found that a dense network approach performs slightly better in terms of higher accuracy, although it uses more parameters and takes a bit longer to train. The best of our convolutional models demonstrated comparable performance with smaller number of parameters and simpler training process. We believe that convolutional approach for analysis of multi-input NLP models is a perspective one and deserves more attention in the future research.

The approach based on convolutions of matrices created by word embeddings has been proven to be fruitful, as presented in [5, 6]. It allows analysis of a single sentence and uses general word embeddings that miss the context, but such methods demonstrate good results on relatively simple tasks, like sentiment analysis (see [8]). Paper [3] is devoted to 1D convolutions based on word order, and [4] contains a detailed discussion on 2D convolutions applied to one-hot word representation. Convolutions applied to word embeddings for a lexical analysis were used in [7, 15], while [11] is devoted to a semantic clustering problem.

An interesting approach to capture the contextual meaning of the words in a pre-transformers era was developed in [9]. The authors built a convolution-based model to obtain a latent semantic representation for a given text. Another work of a similar kind, where convolutions are applied for development of semantic embeddings from hashtags, is [12].

Finally, the paper [13] is devoted to general discussion about application of convolutions in text mining and natural language processing.

The paper is organized as follows. In Section 2 we describe the initial phase of data processing.

We move on to the category prediction in Section 3. The different approaches for decision prediction are considered and compared in Section 4. The conclusion remarks are given in Section 5.

2 Initial Phase

We started with about 700 labeled court cases, each including the data that is described in Table 1:

Table 1: Variables and their description

variable	description	average length
claim	fragment of text describing the essence of a claim	12300
claim justification	fragment of text motivating the claim	12300
response	fragment of text that includes defendant's response	12300
category	categorical parameter describing a type of a case	7 categories in total
decision	decision in the first instance	4 types

Our analysis pursued three goals:
1. to identify whether the categories are balanced and have impact on decision;
2. to predict the category by a claim;
3. to predict the decision by claim, claim justification and response.

We used simple descriptive statistics to examine our categories. We found out that categories are unbalanced, with roughly 55-60% of cases being judged in favour of plaintiff.

The second task (category prediction) is a very important practical problem. The corpus of judicial documents does not include good categorization, while it is essential for such problems. Different types of civil procedure require different sets of documents to submit, have different chances to win the case, and have different durations and associated costs. That is why categorization itself is a crucial problem. We used dense neural network to predict categories.

Finally, the most important goal of this research is to predict the first instance's decision. To do this, we used our three text variables as predictors and developed two families of neural networks: one is based on 2D convolutions and another is based on dense neural networks.

The actual problem that needed to be solved was converting text parameters into numeric vectors. Our text data are in Ukrainian language, so we used a variant of XLM/RoBerta large language model adapted for the Ukrainian language only. This model demonstrates fast inference while sharing the overall performance of the original XLM/Roberta model. We removed the head of the model and used the last layer as our numeric representation, which allowed us to convert all our text data into 768-dimensional numeric vectors used in our further analysis.

3 Category prediction

In order to predict categories we experimented with dense neural networks. There were used claim as the predictor and achieved about 98% accuracy on the test dataset with such model.

On the pie chart (Fig. 1) we can see how many cases belong to the top six categories.

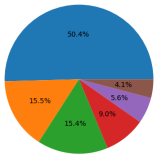

Figure 1: Category's distribution

We can see that the most popular category includes about 50% of all cases, the second and third include about 15%, the fourth is about 9%, and the rest is 5% and below, which makes our categories distribution quite unbalanced.

The model we developed uses *claim* text variable as its input (it is encoded as 768-dimensional numeric vector), which is passed through the two dense layers of 512 neurons each before the final classification layer. We used ReLu activation for dense layers and softmax for the last one.

We trained our model for 20 epochs with cross-entropy loss function, Adam optimizer and learning rate of 10^{-4} and obtained 95% accuracy on a test dataset.

Since our dataset is imbalanced, we evaluated the performance of each category prediction by calculating Precision, Recall and F1 score, which gives the following results.

Table 2: Prediction performance

category	total number	TP	FP	FN	Precision	Recall	F1
1	99	98	8	1	0.92	99	0.96
2	16	16	0	0	1	1	1
3	18	17	1	1	0.94	0.94	0.94
4	34	34	0	0	1	1	1
5	6	6	0	0	1	1	1
6	22	22	0	0	1	1	1
7	2	2	0	0	1	1	1

where TP is True Positive, FP is False Positive and FN is False Negative. Let the category under consideration is category 1, say. Then True Positive means the number of correct predictions, True Negative means number of prediction for 1 category while the true category is different and False Negative means prediction for another category while the true category is 1. Here

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{Total\ Positive},$$

$$Recall = \frac{TP}{TP + FN} =$$

$$= \frac{TP}{Total\ number\ of\ positive\ instances},$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}.$$

We can see that model misclassifies the first category, which is expected having the imbalance of the dataset. At the same time we can see that the model's prediction performance is even (homogeneous) across all categories.

4 Decision prediction

As for the term 'decision', we are using one of the following outcomes from the final judgment of the first-instance court: 'complete satisfaction of the claim', 'decline the claim', 'partially satisfy the claim' or 'other'. For the decision prediction we built and evaluated two families of deep models. One of the area of our interest was evaluation how 2D-convolutions (Conv2D) perform themselves in multi-sentence input setup. Normally, 2D-convolutions are used in computer vision problems, and they are known to be effective in capturing spatial effects, such as identifying an eye or an ear in a certain area of an image. In our case, each element of the numerical representation obtained from the large language model contains some particular information about the sentence. Our motivation was caused by the fact that there is at least vertical spacial alignment between elements of the input matrix. In addition to that, other experiments with convolutions, 1-D convolutions, in particular, demonstrated promising results in reducing the number of parameters required, improving the

training process and learning valuable insights from the original numerical sequences representing inputs.

For this purpose, we developed three models (small, medium and large) based on convolutions and three models based on dense networks. Our goal was to evaluate the performance of convolution-based models, their difficulty to train and approximate dependence of a number of parameters.

It is appeared that decisions of about 60% of all cases in our dataset were "complete satisfaction of the claim". It is worth mentioning that we only used a subset of civil cases related mainly to pension and social aid, that's why the percentage of 'positive' decisions is so high. So, 60% is our baseline.

All Conv2D models take input of the size 3×768. We tried two variants of a loss function: regular cross-entropy loss function and weighted cross-entropy with weights backwards proportional to the class distribution. We repeatedly trained multiple models every time comparing the performance and difficulty of training and evaluating the model's expressiveness. We trained each model with a learning rate of 10^{-3} for some number of epochs (usually between 200 and 400), then selected the best candidates and ran the training for another 100-200 epochs with the smaller learning rate of 10^{-4}.

4.1 Conv2D-small network

Our first Conv2D-small network model has the following architecture:

1. Conv2D layer with 10 3×3 kernels and ReLU activation function, that outputs tensor of the size $1 \times 766 \times 10$.
2. Conv2D layer with a single 1×1 kernel and ReLU activation function, that outputs tensor of the size $1 \times 766 \times 1$.
3. Flatten layer followed by Linear 766×20 layer with ReLU activation.
4. Dropout layer with $p = 0.3$.
5. Final classification layer.

The model has $15,556$ trainable parameters in total. The best achieved result in the sense of accuracy was 0.75, i.e. 15% above the baseline.

We may conclude that such a network does not have enough expressiveness, but at the same time it can be relatively easily trained and does not require much fine-tuning.

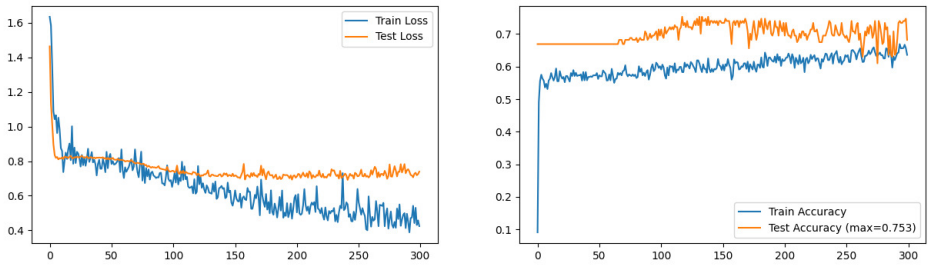

Figure 2: Conv2D-small network model performance

4.2 Conv2D-standard network

The second Conv2D-standard network model has the following architecture:

1. Conv2D layer with 10 3×3 kernels and ReLU activation function, that outputs tensor of the size $1 \times 766 \times 10$.
2. Conv2D layer with a single 1×1 kernel and ReLU activation function, that outputs tensor of the size $1 \times 766 \times 1$.
3. Flatten layer followed by a Linear 766×50 layer with ReLU activation.
4. Dropout layer with $p = 0.3$.
5. Another 50×30 linear layer with ReLU activation.
6. Dropout layer with $p = 0.3$.

7. Final classification layer.

The model has 40146 trainable parameters in total. The best achieved accuracy result was 0.786, or 18.6% above the baseline.

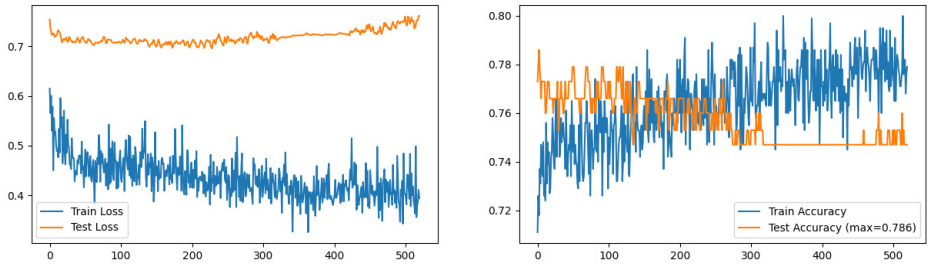

Figure 3: Conv2D-standard network model performance

The best result has been achieved with the first training attempt (without the fine tuning). However, we find that models with weights require some fine-tuning, while standard models usually achieve their best results during the first training attempt.

4.3 Conv2D-deep network

Our last Conv2D-deep network model has the following architecture:

1. Conv2D layer with 10 3×3 kernels, padding="same" and ReLU activation function, that outputs tensor of the size $1 \times 766 \times 10$.
2. Another Conv2D layer with 10 3×3 kernels, and ReLU activation function, that outputs tensor of the size $1 \times 766 \times 10$.
3. Conv2D layer with a single 1×1 kernel and ReLU activation function, that outputs tensor of the size $1 \times 766 \times 1$.
4. Flatten layer followed by a Linear 766×30 layer with ReLU activation.
5. Dropout layer with $p = 0.3$.
6. Another 30×20 linear layer with ReLU activation.
7. Dropout layer with $p = 0.3$.
8. One more 20×30 linear layer with ReLU activation.
9. Dropout layer with $p = 0.3$.
10. Final classification layer.

The model has $25,436$ trainable parameters in total. The best achieved accuracy result was 0.73, i.e. 13% above the baseline.

We may conclude that such a network does not have enough expressiveness, but at the same time is relatively easy to train and does not require much fine-tuning.

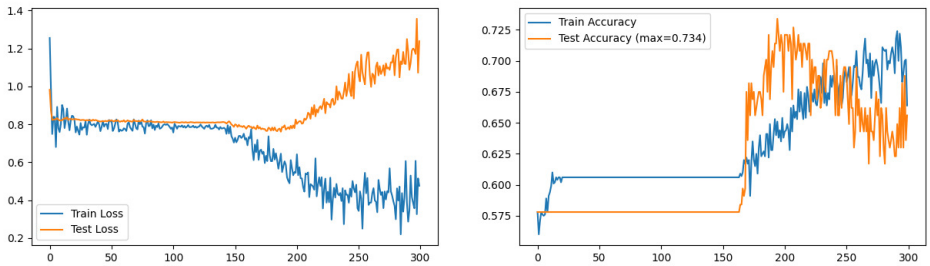

Figure 4: Conv2D-deep network model performance

4.4 Dense-small network

Our first Dense-small network model has the following architecture:

1. Flatten layer that converts (3×768) inputs into one 2304-dimensional array.
2. Dense layer that outputs 20 neurons followed by ReLU activation and Dropout.
3. Final classification layer.

The model has $46,205$ trainable parameters in total. The best achieved accuracy result was 0.818, i.e. $21,8\%$ above the baseline. This result has been achieved after multiple iterations of training each consisting of 100-200 epochs and reducing learning rate scheme.

We can see that "small" network has much more parameters and convolution-based networks and renders better accuracy. However it takes more time to converge and is more difficult to train.

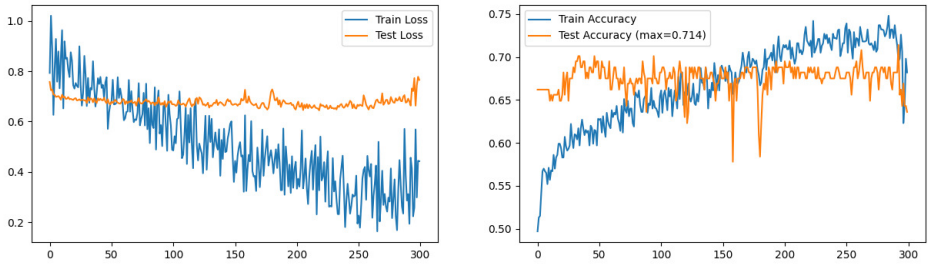

Figure 5: Dense-small network model performance (1st series of 300 epochs)

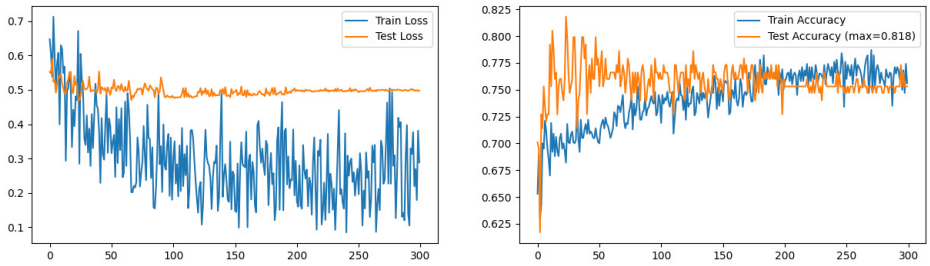

Figure 6: Dense-small network model performance (2nd series of 300 epochs)

4.5 Dense-medium network

The second Dense-medium network model has the following architecture:

1. Flatten layer that converts (3×768) inputs into one 2304-dimensional array.
2. Dense layer that outputs 50 neurons followed by ReLU activation and Dropout.
3. Another dense layer that outputs 30 neurons followed By ReLU activation and Dropout.
4. Final classification layer.

The model has $116,935$ trainable parameters in total. The best achieved accuracy result was 0.857, or $25,7\%$ above the baseline.

This is the best result we recorded. The training process was relatively involving, required multiple learning rates adjustment and testing different combinations of batch size and other parameters. This result was achieved using a weighted loss function.

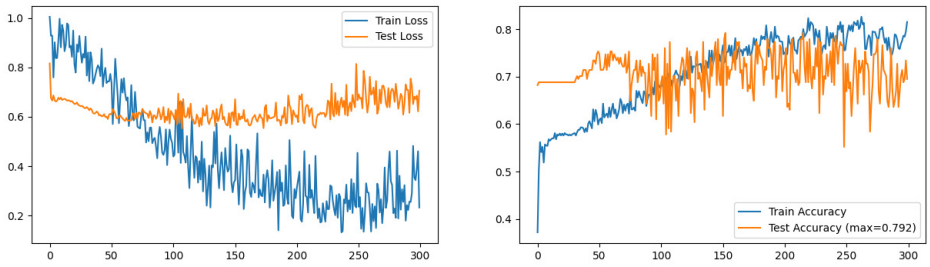

Figure 7: Dense-medium network model performance (1st series of 300 epochs)

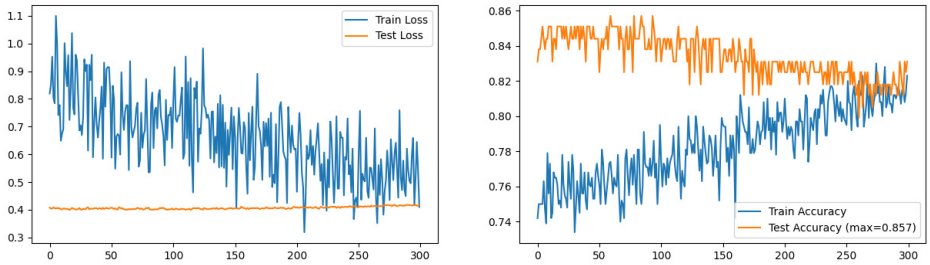

Figure 8: Dense-medium network model performance (2nd series of 300 epochs)

4.6 Dense-large network

Our last Dense-large network model has the following architecture:

1. Flatten layer that converts (3×768) inputs into one 2304-dimensional array
2. Dense layer that outputs 100 neurons followed by ReLU activation and Dropout
3. Another dense layer that outputs 50 neurons followed By ReLU activation and Dropout
4. Yet another dense layer that outputs 30 neurons followed By ReLU activation and Dropout
5. Final classification layer

The model has $237,235$ trainable parameters in total. The best achieved accuracy result was 0.786, i.e. $18,6\%$ above the baseline.

Despite all adjustments and experiments we can clearly see that this model overfits the data and demonstrates the worse performance of all dense networks.

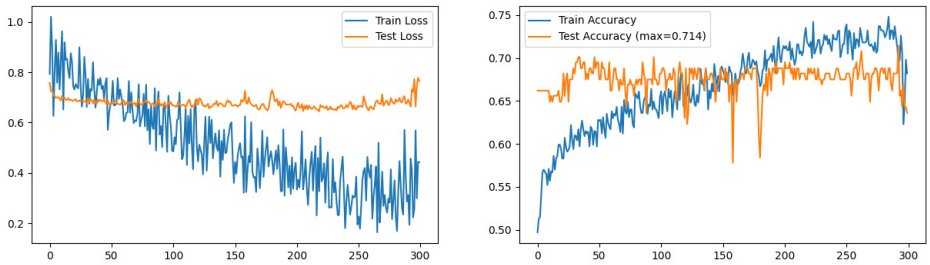

Figure 9: Dense-large network model performance (1st series of 300 epochs)

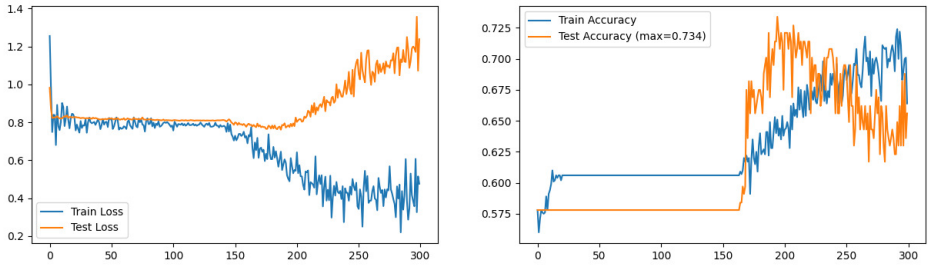

Figure 10: Dense-large network model performance (2nd series of 300 epochs)

5 Conclusions

In this paper, we have developed the algorithm for classification of various parameters related to the civil procedure, such as court case category and first instance decision. One important feature of our algorithm is the use of multiple text inputs such as court claim, claim motivation and defendant's response. In order to process such inputs, we used XLM/RoBerta language model, which provided us with numerical representations for such text inputs. Thus, we obtained multiple 768-dimensional vectors that represent our input. In order to process them all at the same time, we tried multiple approaches. One approach was to use convolutional layers, typically used in computer vision models, to process multiple layers at once. Our motivation was that each numerical representation keeps the same type of information at the same position of the embedding obtained from the LLM, and so there is some spatial structure in our inputs if we represent them as a matrix.

As alternative to convolutions, we used dense neural networks consisting of fully connected layers. We tried multiple architectures of each type and searched for the best combination of hyperparameters. Finally, we arrived to the following results.

Dense networks, in general, produce better quality (measured by accuracy). However, they are more difficult and expensive to train. Convolutional networks are usually easier to train, and they also produce good results, although slightly worse than that of dense networks.

We may conclude that the use of convolutions in such tasks deserves further investigation. It may be fruitful to try to reshape the output of a large language model so that the similar features are located close to each other in the resulting numerical vector, thus increasing the spatial connectivity of inputs.

References

1. DEVLIN, J., CHANG, M., LEE, K., and TOUTANOVA, K. (2019) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171-4186, Available from: https://aclanthology.org/N19-1423/

2. GOLOMOZIY, V., MISHURA, Y., IZAROVA, I., and IANEVYCH, T. (2023) Processing Big Data of Court Decisions, *BALTIC JOURNAL OF MODERN COMPUTING*, Vol. 11, No. 4. Available from: https://www.bjmc.lu.lv/contents/

3. JOHNSON, R., and ZHANG, T. (2015) Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 103-112. Available from: https://aclanthology.org/N15-1011/

4. JOHNSON, R., and ZHANG, T. (2015) Semi-supervised Convolutional Neural Networks for Text Categorization via Region Embedding. In *Advances in Neural Information Processing Systems* 28, pp. 919-927.

5. KALCHBRENNER, N., GREFENSTETTE, E., and BLUNSOM, P. (2014) A Convolutional Neural Network for Modelling Sentences.

In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers), pp. 655-665. Available from: https://aclanthology.org/P14-1062/

6. KIM, Y., (2014) Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1746-1751.

7. NGUYEN, T. H., and GRISHMAN, R. (2015) Relation Extraction: Perspective from Convolutional Neural Networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pp. 39-48. Available from: https://aclanthology.org/W15-1506/

8. SANTOS, C. N. DOS, and GATTI, M. 2014. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pp. 69-78. Available from: https://aclanthology.org/C14-1008/

9. SHEN, Y., HE, X., GAO, J., DENG, L., and MESNIL, G. (2014) A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pp. 101-110. https://doi.org/10.1145/2661829.2661935

10. SONI, S., CHOUHAN, S.S. and RATHORE, S.S. (2023) TextConvoNet: a convolutional neural network based architecture for text classification. *Appl Intell* 53, 14249-14268. https://doi.org/10.1007/s10489-022-04221-9

11. WANG, P., XU, J., XU, B., LIU, C., ZHANG, H., WANG, F., and HAO, H. (2015) Semantic Clustering and Convolutional Neural Network for Short Text Categorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 352-357. Available from: https://aclanthology.org/P15-2058/

12. WESTON, J., and ADAMS, K. (2014) TagSpace: Semantic Embeddings from Hashtags. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing.*, pp. 1822-1827. Available from: https://aclanthology.org/D14-1194/

13. WIDIASTUTI, N., (2019) Convolution Neural Network for Text Mining and Natural Language Processing. *IOP Conference Series: Materials Science and Engineering*, Issue 5, Vol. 662. doi: 10.1088/1757-899X/662/5/052010

14. YUAN, K., GUO, S., LIU, Z., ZHOU, A., YU, F., and WU, W. (2021) Incorporating Convolution Designs Into Visual Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.*, pp. 559-568. doi: 10.1109/ICCV48922.2021.00062

15. ZENG, D., LIU, K., LAI, S., ZHOU, G., and ZHAO, J. (2014) Relation Classification via Convolutional Deep Neural Network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pp. 2335-2344. Available from: https://aclanthology.org/C14-1220/

Received: 28.10.2023

The work has been done within the project "Innovative technologies for processing court decisions using machine learning algorithms" K-I-186, financed by an external instrument of assistance of the European Union to fulfill Ukraine's obligations in the European Union Framework Program for Research and Innovation "Horizon 2020".