International Journal of User-System Interaction, 13(4) 2020, 183-198

Intelligent Linguistic System for the Grammar of the Romanian Language

Ioan Florin Cătălin Nițu, Traian Eugen Rebedea

University Politehnica of Bucharest, Faculty of Automatic Control and Computer Science
313 Splaiul Independenței, Bucharest 060042
E-mail: ioan_florin.nitu@stud.acs.upb.ro, traian.rebedea@cs.pub.ro

Abstract. The field of natural language processing is not as strongly developed for the Romanian language as it is for others, as is the English language. Writing texts correctly has always been a necessity, and the development of tools that will be useful in this need is critical. The proposed correction system receives a sentence with grammatical errors and corrects it, using state-of-the-art technologies to perform this operation such as attention-based neural models based on Encoder-Decoder Transformers. These are a cornerstone in the development of intelligent tools for processing – translating, summarizing, or proofreading – texts and are the foundation for this project. The paper uses RONACC, the first corpus for grammatical corrections in Romanian for modeling, training, testing, and validating the project. Using a very large dataset with over a million learning examples, an average BLEU score of 45.29 points was obtained, in a rather short training time executed on several GPUs. However, even a smaller dataset of only fifty thousand examples with as many as one hundred epochs achieves an average BLEU score of 33.29 points in three hours.

Keywords: Romanian language, grammar, transformers, attention, encoder-decoder

DOI: 10.37789/ijusi.2020.13.4.1

Introduction

Grammar (from ancient Greek, γραμματική) is an important part of linguistics and represents the set of structural rules that establish in a natural language the relationships for the composition of clauses, sentences, and words; but it also aims at the study of these rules, including phonology, morphology, and syntax, but also phonetics, semantics, and pragmatics.

Romanian is the official language of Romania (implicitly one of the official languages of the European Union) and the Republic of Moldova (also referred to as the Moldovan language). Romanian is part of the sub-branch of the Eastern Romance languages of the Neo-Latin languages – a linguistic group that developed from dialects of Vulgar Latin, separate from the

Western Romance languages between the 5th and 8th centuries – as part of the Balkan Romance languages (Romanian dialects): in addition to Romanian (Daco-Romanian), being also Aromanian (Macedonian-Romanian), Istro-Romanian and Megleno-Romanian. The Romanian language divisions are sub-dialects (accents or graiuri), grouped as Nordic (Moldovan, Transylvanian, Banat) and southern (Muntenian, Dician).[1]

In Romania, the rate of functional illiteracy is 39% among students[2], and grammatical errors are frequently in all population categories[3]. For English, various programs, like Grammarly[4], have appeared that help users write correctly, intelligibly, and clearly, so that the message sent is rich in information, concise, also easy to navigate. The Romanian language does not have such a computer program. As society moves at an increasingly fast pace, people need quick and efficient solutions for their problems.

The system proposed in this paper uses state-of-the-art technologies and mechanisms to create a correcting system for the grammar of the Romanian language based on neural networks architecture. The results obtained in terms of the BLEU score are quite impressive given the limited training resources and time.

Existing Approaches

Correcting texts and natural language, in general, is often encountered in computer science, artificial intelligence, and machine learning. The study of natural language processing has brought technologies, models, and products

---

[1] *Romanian language*. (n.d.). Retrieved October 9, 2019, from Wikipedia: https://en.wikipedia.org/wiki/Romanian_language

[2] *39% dintre elevii români sunt analfabeți funcțional. Cum îi va afecta pe viitor.* (2019, January 24). Retrieved October 9, 2019, from Digi24: https://www.digi24.ro/stiri/actualitate/educatie/39-dintre-elevii-romani-sunt-analfabeti-functional-cum-ii-va-afecta-pe-viitor-1070140

[3] Paraschivescu, R. (2018, July 26). Cei care vorbesc corect vor forma un soi de trib, de sectă, care se va refugia undeva în păduri. *Contributors*. (L. Popescu, & A. Radulescu, Interviewers) Retrieved June 16, 2020, from Contributors: http://www.contributors.ro/cultura/interviu-radu-paraschivescu-”cei-care-vorbesc-corect-vor-forma-un-soi-de-trib-de-secta-care-se-va-refugia-undeva-in-paduri”

[4] *Grammarly*. Retrieved June 20, 2020: https://www.grammarly.com

that can competitively address this issue.

For learning tasks such as machine translation or automatic text correction, different methods had been proposed. One such approach uses Deep Neural Networks with multilayered long short-term memory to map input sequences to a fixed-size vector or to decode the output (Sutskever, Vinyals, & Le, 2014). Other architectures comprise two recurrent neural networks trained together that act like an Encoder-Decoder (Cho, et al., 2014). To overcome a deadlock in performance improvements in the fixed-length vector approach, an extended method is to autonomously search for significant bits of an input sentence to predict a word (Bahdanau, Cho, & Bengio, 2014). In contrast to recurrent models, another design is solely built on convolutional neural networks having the advantage that during training, calculations can be entirely parallelized for all elements. Moreover, this architecture has an attention component installed in every decoding layer (Gehring, Auli, Grangier, Yarats, & Dauphin, 2017). Built exclusively with attention mechanisms, the transformer architecture is a simple network without any recurrences and convolutions. These are superior in terms of quality, parallelizable, and taking much less training time, as revealed by two experiments for tasks on machine translation. Also, the transformer generalizes well in text analysis and text generation (Vaswani, et al., 2017).

In the Romanian language, several systems were proposed for text correction, most of them using rule-based approaches. Florea et al. (2020) proposed a multi-layer rule-based system to enforce some of the formal writing rules in Romanian while Iamandi (2017) used rule-based correction with a full form lexicon for morphological analysis. Coteț et al. (2020) used neural Transformer architectures for Romanian grammatical error correction and introduced a novel corpus, RONACC[5].

Technologies Used

This paper uses a Transformer-based solution, as proposed by Vaswani, et al. (2017), to implement a grammatical correction system.

In sequence modeling and reasoning challenges, like machine translation

[5] RONACC (2020). Romanian National Audiovisual Council Corpus [Dataset]. Retrieved June 10, 2020, from: https://nextcloud.readerbench.com/index.php/s/9pwymesT5sycxoM

and language modeling, recurrent neural networks and long short-term memory have positioned themselves as state-of-the-art techniques (Cho, et al., 2014) (Bahdanau, Cho, & Bengio, 2014) (Vaswani, et al., 2017). Up until now, there have been many endeavors that tried to expand the limits of encoder-decoder architectures and recurrent language models (Wu, et al., 2016) (Vaswani, et al., 2017).

In different tasks, attention mechanisms for reasoning and persuasive sequence models are an indispensable component that allows dependencies modeling without considering their distance in input or output sequences (Bahdanau, Cho, & Bengio, 2014) (Vaswani, et al., 2017). In most cases, a recurrent network is combined with attention mechanisms. To avoid recurrences, transformers are introduced as a model architecture that attracts global dependencies, using a self-attention mechanism, linking the input and output. Admitting a notably higher parallelization, the transformer can extend into a new stage in terms of the texts' quality of translation or correction after training (Vaswani, et al., 2017).

Proposed Solution

Transformers are incredible models based on an encoder-decoder type structure (Sutskever, Vinyals, & Le, 2014) (Cho, et al., 2014) (Bahdanau, Cho, & Bengio, 2014) (Gehring, Auli, Grangier, Yarats, & Dauphin, 2017) (Vaswani, et al., 2017), taking the world of natural language by storm. They pushed the barriers of state-of-the-art technology and broke records in the NLP world. They have found their place in various applications, from machine translation and conversational agents to the improvement of search engines, and are the latest craze in deep learning (Phi, 2020).

Following are presented their theoretical aspects and how they are used in natural language processing. The proposed solution uses transformers to process natural language and to correct texts written in Romanian.

Competitive neural reasoning models use an encoder-decoder type structure (Vaswani, et al., 2017). The encoder transforms the symbolic input $x = (x_1, \ldots, x_n)$ into a sequence of continuous representations $z = (z_1, \ldots, z_l)$. With z, the decoder generates the symbolic output $y = (y_1, \ldots, y_m)$, one element after another (Vaswani, et al., 2017). The model is auto-regressive: using the previously predicted elements as an additional input for predicting

the next element (Graves, 2013). The transformer is based on this architecture using self-attention and fully connected point-wise layers for the encoder and decoder (Vaswani, et al., 2017).

Self-attention

Self-attention is a mechanism by which transformers can have infinite theoretical memory and focus on words generated long before (Vaswani, et al., 2017).

The entry is distinguished into three layers to create the query, key, and value vectors to gain attention (Phi, 2020). The attention calculation is performed according to Vaswani, et al. (2017). Multi-headed attention involves the linear design of queries, keys, and values with different learned linear projections. Each one will be the input for the attention computation, and the calculations can be performed in parallel, the results being concatenated at the end (Vaswani, et al., 2017).

Transformer

The transformer is composed of an encoder (with N encoding layers), a decoder (with N decoding layers), and a linear final layer the size of the target vocabulary, as in Figure 1 (Vaswani, et al., 2017).

The encoder maps all input sequences into continuous abstract representations, containing the learned information about the entire sequence. The decoder is auto-regressive, starts with a start token, and receives the previous outputs as inputs along with the encoder outputs that contain information about the input attention (Phi, 2020). The output of the decoder passes through a final linear layer acting as a classifier (the number of classes is equal to the size of the vocabulary), and the predicted word is given by:

$$predicted_{id} = \underset{id \in \{0 \ldots vocab_{size}\}}{\operatorname{argmax}} \; softmax(id)$$

The obtained output is added to the decoder inputs and the procedure continues until the final token is reached (Phi, 2020).

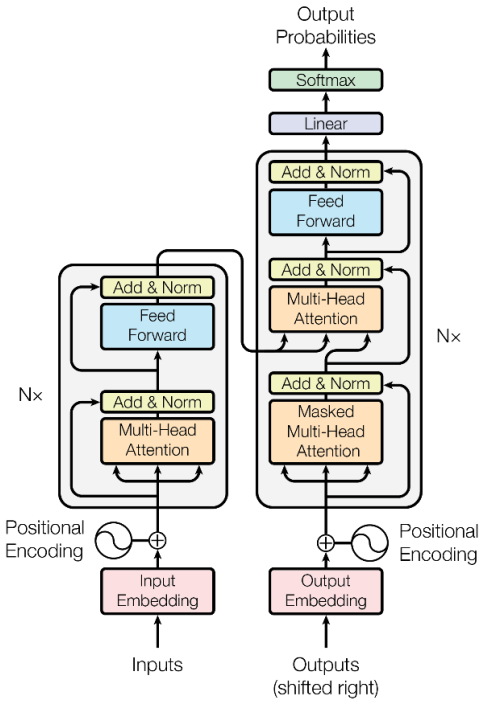


Figure 1. Transformer model (Vaswani, et al., 2017, p. 3)

Implementation Details

The implementation of the solution was done in a Jupyter Notebook[6], using Google Colab[7] for development. The code was written in Python3[8] using the TensorFlow[9] framework for model development and training and TensorFlow Datasets[10] for dataset manipulation. The NumPy[11] library was used for mathematical calculations and operations, besides TensorFlow. Also, the Matplotlib[12] library was used for graphical representations.

---

[6] *Jupyter Notebook*. Retrieved October 01, 2019: https://jupyter.org
[7] *Google Colab*. Retrieved February 20, 2020: https://colab.research.google.com
[8] *Python*. Retrieved June 03, 2020: https://www.python.org
[9] *TensorFlow*. Retrieved June 03, 2020: https://www.tensorflow.org
[10] *TensorFlow Datasets*. Retrieved June 03, 2020: https://www.tensorflow.org/datasets
[11] *Numpy*. Retrieved June 03, 2020: https://numpy.org
[12] *Matplotlib*. Retrieved June 03, 2020: https://matplotlib.org

Configuring the Input Pipeline

The dataset consists of several TSV (Tab-Separated Values) files, each line in the file containing a pair of shapes (correct sentence, wrong sentence), where the correct sentence represents the expected reference or result in following the correction of the incorrect sentence. The datasets, although divided into several files, are concatenated during running and are organized into three categories:

- Training/learning datasets;
- Validation datasets;
- Test datasets.

After loading the datasets, two tokenizers are used (one for the wrong sentences and the other for the correct sentences) using Subword Text Encoders. Text encoders of this type are constructed from a corpus of sentences from which sub-words will be generated. The result is a mapping of the words identified in the text to integers. These numbers can be used to encode words into numbers or to decode numbers back, respectively.

To train, validate, and test the model, all datasets must be encoded using the two tokenizers (to encode both the wrong sentences and the correct sentences). Besides, each sentence will have two extra tokens specially assigned:

- start token (sos: Start Of String);
- end token (eos: End Of String).

Each encoder is composed of a vocabulary with a size specified in the construction. Each word has a corresponding numerical identifier in the dictionary, smaller than the vocabulary size. Therefore, the values assigned to the two extra tokens must be chosen greater than or equal to the vocabulary size.

Words for which there was no vocabulary association were split into sub-words for which there was an association, or new associations were created where they did not exist. The vocabulary was built using the dataset.

Dataset

The dataset used to train, validate and test the model is the RONACC corpus. The initial dataset consisted of pairs of lines, as in Table 1, where the first line was the correct one and the second the wrong one (containing grammatical errors or misspellings). It was preprocessed so that the pairs

were on the same line (facilitate a load of data easier to achieve) separated using the tab character ("\t" for TSV files). When reading from the file, the data is interpreted using the "\t" separator, and the corresponding datasets of the form (wrong sentence, correct sentence) are created. In Table 1, the mistakes were highlighted along with the right variants with italics.

Table 1 – Samples from the dataset

Correct Sentence	Wrong Sentence
Acoperișul din șindrilă nu se mai păstrează, fiind înlocuită cu țiglă în 1936, fapt ce a necesitat sprijinirea acoperișului cu structuri *improvizate*.	Acoperișul din șindrilă nu se mai păstrează, fiind înlocuită cu țiglă în 1936, fapt ce a necesitat sprijinirea acoperișului cu structuri *inprovizate*.
Alte mărci comerciale utilizate vreme îndelungată sunt Löwenbräu, *deținătorii* căreia spun că este folosită din 1383, și Stella Artois din 1366	Alte mărci comerciale utilizate vreme îndelungată sunt Löwenbräu, *deținători* căreia spun că este folosită din 1383, și Stella Artois din 1366.
Cea mai importantă este *cea* surprinsă asupra *lunii* Noiembrie.	Cea mai importantă este *ceea* surprinsă asupra *luni* Noiembrie.

Masking

The masks used are those for filling and those for looking ahead. Fill masks ensure that the model will not treat the filling as input. They indicate where there are zero values (TensorFlow, 2019).

The masks with look-ahead are used to indicate which tokens should not be used. For example, to predict the word *i*, only words 1, 2, …, *i* - 1 will be used, so the model is not allowed to see the expected output. For this, a triangular lower matrix of size $i \times i$ is created which is then complemented to a triangular upper matrix without the main diagonal (TensorFlow, 2019).

The function of creating the masks deals with the encoding (filling) and decoding (filling) ones used in the second attention block of the decoder to mask the outputs of the encoder, but also a combined mask (filling mask and masking mask with look-ahead) used in the first decoder attention block both for future input tokens received by the decoder. The first two masks are created relative to the wrong sentence, and the combined one dependent on the correct sentence.

Training

Training is done in several epochs. From time to time, after some epochs, a

checkpoint is made to save the model created by the training process.

The transformer is an autoregressive model (Graves, 2013): it makes predictions one by one and then decides, based on its previous output, what should be the next output (TensorFlow, 2019).

This model employs teacher-forcing throughout training. Teacher-forcing, despite the model's prediction regarding the present step, puts the correct result into the following step (TensorFlow, 2019). For this, two sentences are created from the correct sentence:

- The first sentence (the correct input sentence) does not have the last word;
- The second sentence (the correct reference sentence) does not have the first word; thus, for each index in the first sentence, the predicted one is found in the second sentence.

To achieve a better prediction of the next word, the transformer, through attention, makes use of the previous words from the input sequence (TensorFlow, 2019).

The masks are created using the wrong sentence and the correct input sentence. Then, predictions are obtained using the transformer, and the loss between the correct reference sentence and the result is calculated. The gradients are computed, and then the optimizer is applied. It saves training loss and accuracy between the correct reference sentence and prediction. Dropout is used to avoid overfitting.

For each epoch, the values for loss and training accuracy are reset. In each era, one batch from the training set is taken one by one, and then a training step is taken. Loss and accuracy statistics are displayed after a certain number of batches. At the end of an epoch, loss, accuracy, and training time are displayed. To plot these metrics during training, TensorBoard[13] visualization tool was used.

Results Evaluation

The model was evaluated on a test set with 1,519 pairs of sentences and several training sets: small - 7,082 pairs of sentences; medium - 7,082 +

[13] *TensorBoard*. Retrieved June 03, 2020: https://www.tensorflow.org/tensorboard

50,000 pairs of sentences; large - 7,082 + 1,000,000 pairs of sentences.

The model was trained for 100 epochs on the (small and) medium dataset, at each epoch testing and measuring accuracy (Figure 2) and loss (Figure 3).

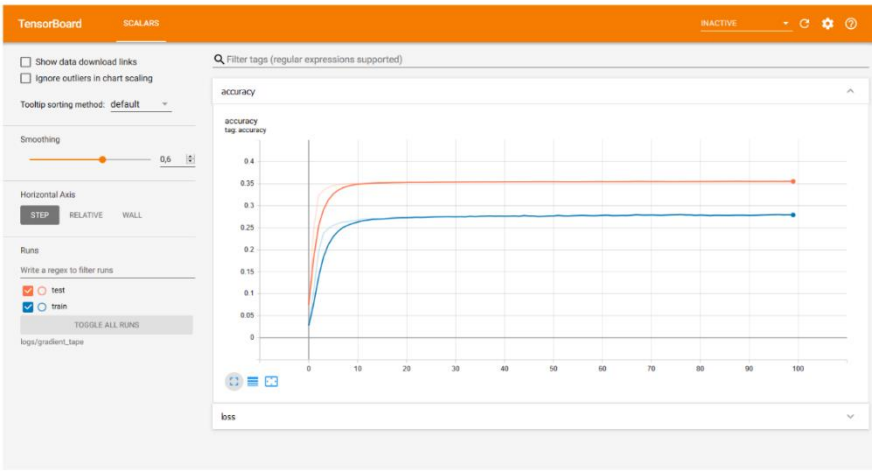


Figure 2. Accuracy for the medium dataset

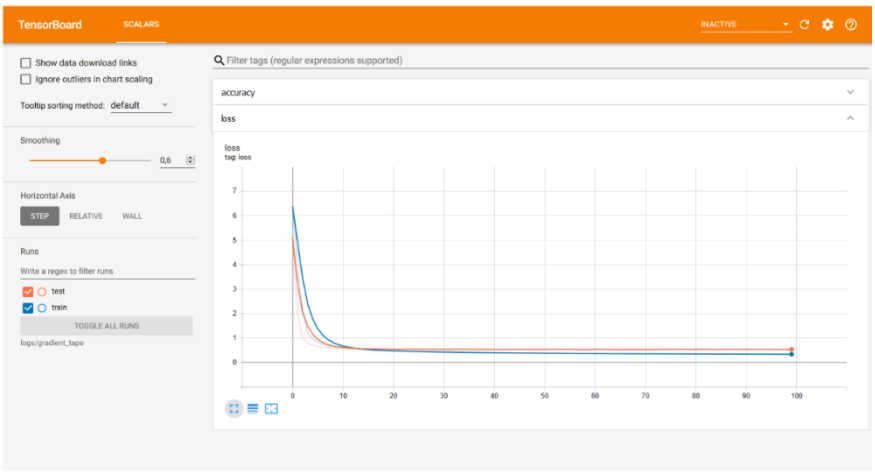


Figure 3. Loss for the medium dataset

In the end, the results were validated using a validation set of 1,518 pairs of sentences. From the graphs, it can be seen how these values converge, and also, the loss tends to zero.

The model proposed by Vaswani, et al. (2017) obtains a BLEU score of

27.3, respectively 38.1 for English to German translation, respectively French for the Transformer with the base model. For the big Transformer, a BLEU score of 28.4, respectively 41.8. The model implemented in the presented work obtained a competitive BLEU score of 20.56 for text corrections in Romanian with a modest training set. With an increased training set, it got a score of 33.29, as shown in Table 2, representing a substantial improvement.

The data can not be directly compared, as the article by Vaswani, et al. (2017) used other datasets, was applied on different problems, and with different languages. However, it should be noted that the field is currently quite limited for correcting texts in Romanian to make comparisons on other systems. In the below table, it can be seen how an increased set of training data can substantially influence the score (45.29 BLEU score for the large training set).

Table 2 – Training results

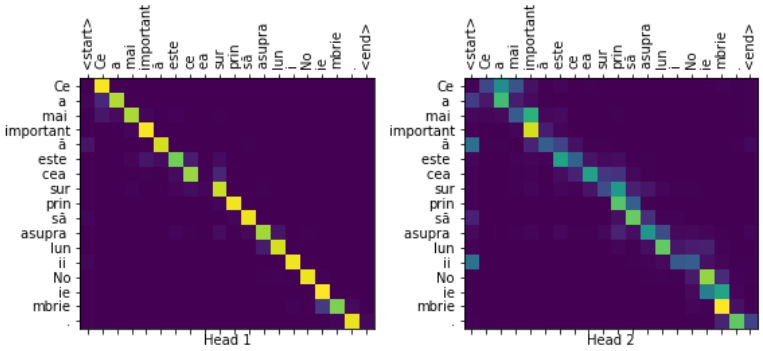
Dataset	Epochs	BLEU Score	Training time
Small	100	20.56	9.66 s / epoch
Medium	100	33.29	99 s / epoch
Large	5	**45.29**	1032 s / epoch

The calculation of the BLEU score (Papineni, Roukos, Ward, & Zhu, 2002) was implemented using nltk.translate.bleu_score.sentence_bleu and nltk.translate.bleu_score.SmoothingFunction with method4 from NLTK[14] for the case of short sentences. Another important metric is the attention that words pay to other words, as presented in the examples in Figure 4.

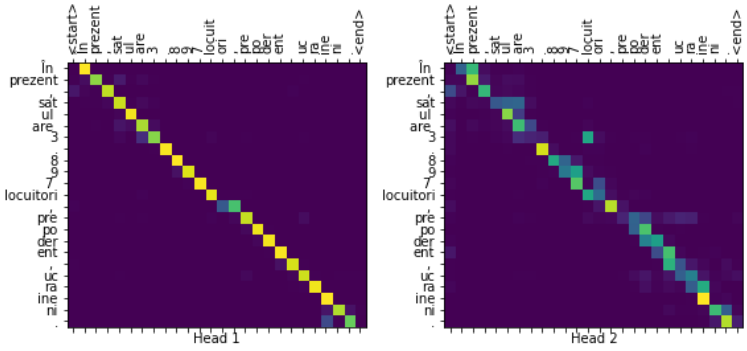
Qualitative Evaluation

In Figure 4, some examples of running the model on different input sentences are presented. Attention was measured at the output (second substrate) of the last layer of the decoder. The examples presented in the below figure are fully corrected cases (a), partially corrected sentences (b), and sentences with errors or that could not be corrected (c).

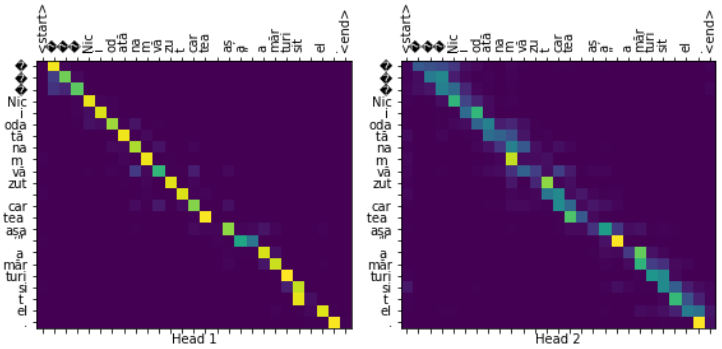
[14] Team NLTK. (2001). *NLTK documentation*, 3.5. (Natural Language Toolkit) Retrieved June 21, 2020, from Natural Language Toolkit: https://www.nltk.org/



(a) BLEU score: 1.0000



(b) BLEU score: 0.7071



(c) BLEU score: 0.2741

Figure 4. Attention examples for pairs of wrong and correct sentences from the dataset

If the quantitative results obtained were on average satisfactory, in terms of the BLEU score, for example, for which improvements were observed, qualitatively, things are slightly different due to the particularities of each type of grammatical error and the language.

An example that the model can easily correct is the case of **ceea – cea**, as is the case with the sentence *"Cea mai importantă este **ceea** surprinsă asupra luni Noiembrie."* for which the system produces the prediction *"Cea mai importantă este **cea** surprinsă asupra lunii Noiembrie."* which is also the correct variant expected at the output. Instead, the case **n-am – nam (lack of hyphen)** it is not as well learned by the model, for example *"„**Nici odată nam** văzut cartea așa" a mărturisit el."* for which the model does not reach the correct variant *"„**Niciodată n-am** văzut cartea așa", a mărturisit el."* being a more complicated case because one lexical part is misspelled (by **nam** instead of **n-am**) and another is misspelled by parting (**nici odată** instead of **niciodată**). The problem of correction also comes from the intersection of the two cases.

Another example for which the model does not generalize well is that of words it does not know (they do not appear in the output dictionary) and does not know how to make associations. For example, in the case of misspelled words, if they have not been learned, they will not get corrected.

For the sentence "*În prezent, satul are 3.897 locuitori, **prepoderent** ucraineni.*" the model will not produce the expected output "*În prezent, satul are 3.897 locuitori, **preponderent** ucraineni.*", because the model does not know the association **prepoderent – preponderent** and does not know that there is a mistake of this type. For this example, the BLEU score is misleading, somewhere around 70%, which may seem to be a fair result.

Such cases must be considered when evaluating the model. Possible solutions to such problems are to increase the number of training epochs and to use training sets (with the disadvantage of increased training time) with an extensive number of learning examples to cover different cases and mistakes frequently encountered in the language.

However, it should be noted, that for a sentence that is spelled correctly, the model will most likely not try to change it, as the words are found in the dictionary, and the links between them produce large weights of attention. More correction examples are listed in Table 3, of both fully corrected sentences and sentences with no corrections.

Table 3 – Examples of corrections made by the proposed model

Wrong Sentence	**Predicted Sentence**	**Correct Sentence**	**BLEU score**
Cea mai importantă este **ceea** surprinsă asupra luni Noiembrie.	Cea mai importantă este **cea** surprinsă asupra lunii Noiembrie.	Cea mai importantă este **cea** surprinsă asupra lunii Noiembrie.	1.0000
Indicele ICI ierarhizează țările **comform** cu performanța lor generală și oferă scoruri pe baza unor piloni și subindici care dau o **ide** generală a performanțelor în domeniile respective.	Indicele ICI ierarhizează țările **conform** cu performanța lor generală și oferă scoruri pe baza unor piloni și subindici care dau o **idee** generală a performanțelor în domeniile respective.	Indicele ICI ierarhizează țările **conform** cu performanța lor generală și oferă scoruri pe baza unor piloni și subindici care dau o **idee** generală a performanțelor în domeniile respective.	1.0000
Fiul său, **dintro** căsătorie morganatică, a fost Franz von Meran.	Fiul său, **dintr-o** căsătorie morganatică, a fost Franz von Meran.	Fiul său, **dintr-o** căsătorie morganatică, a fost Franz von Meran.	1.0000
Epidemia de gripă din 2009 **sa** declanșat, conform comunicatului mexican la OMS, la 18 martie, și nu se știe când **v-a** atinge apogeul.	Epidemia de gripă din 2009 **s-a** declanșat, conform comunicatului mexican la OMS, la 18 martie, și nu se știe când **va** atinge apogeul.	Epidemia de gripă din 2009 **s-a** declanșat, conform comunicatului mexican la OMS, la 18 martie, și nu se știe când **va** atinge apogeul.	1.0000
Omul negru poate **fii** folosit metaforic pentru a exprima o persoană sau un lucru, pentru care cineva are o frică irațională.	Omul negru poate **fi** folosit metaforic pentru a exprima o persoană sau un lucru, pentru care cineva are o frică irațională.	Omul negru poate **fi** folosit metaforic pentru a exprima o persoană sau un lucru, pentru care cineva are o frică irațională.	1.0000
În primul rând, desigur, un buton de empatie va permite Facebook să vă înțeleagă mai bine.	În primul rând, desigur, un buton de empatie va permite Facebook să vă înțeleagă mai bine.	În primul rând, desigur, un buton de empatie va permite Facebook să vă înțeleagă mai bine.	1.0000 (without errors)
„**Nici odată nam** văzut cartea așa” a mărturisit el.	„**Nici odată nam** văzut cartea așa” a mărturisit el.	„**Niciodată n-am** văzut cartea așa”, a mărturisit el.	0.2741
Chestiunea e că eu nu sunt **deacord** cu redenumirea.	Chestiunea e că eu nu sunt **deacord** cu redenumirea.	Chestiunea e că eu nu sunt **de acord** cu redenumirea.	0.5912
În prezent, satul are 3.897 locuitori, **prepoderent** ucraineni.	În prezent, satul are 3.897 locuitori, **prepoderent** ucraineni.	În prezent, satul are 3.897 locuitori, **preponderent** ucraineni.	0.7071
„Participarea nu este semnificativ **diferite** față de anul trecut**”** spune McComb.	„Participarea nu este semnificativ **diferite** față de anul trecut**”** spune McComb.	„Participarea nu este semnificativ **diferită** față de anul trecut**”,** spune McComb.	0.3782

Conclusions

Compared to the model proposed by (Vaswani, et al., 2017), which was trained twelve hours on eight P100 GPUs, the implemented model was prepared on the medium dataset for almost three hours and on the large set for an hour and a half (only five epochs). Only one super dataset was used for modeling, and no comparisons could be made with other datasets. Unfortunately, the field development did not make it possible to compare the implemented model with other models. Furthermore, this implementation is a starting point in field development. It can be used in the construction and modeling of high-performance competitive technologies for text corrections in Romanian. The generalization of the model is not as expected, but there are cases in which it has managed to accurately predict a correct sentence starting from one that had grammatical errors (Figure 4 (a)).

The development and improvement of the model (or at least of the field of text correction in Romanian) are urgently needed, and this can be done using more training data, a larger number of training epochs, or an extensive set of validation data. As technology evolves, the model needs to be updated and improved with the latest technologies. An example of such technology is BERT (Bidirectional Transformer Encoder Representations) (Devlin, Chang, Lee, & Toutanova, 2014).

In addition to improving the model, further application developments require the use of an interface for user-program interaction. The solution code is open-source[15] and uses TensorFlow's Transformer implementation (2019).

Acknowledgments

We would like to thank Teodor Mihai Coteț for all of his help and support.

References

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*. Retrieved June 20, 2020, from https://arxiv.org/abs/1409.0473

[15] The implementation is available at: https://github.com/nitu-catalin1998/diploma-project.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078*. doi:10.3115/v1/D14-1179

Coteț, T. M., Rușeți, Ș., & Dascălu, M. (2020). Neural Grammatical Error Correction for Romanian. *2020 IEEE 32nd International Conference on Tools with Artificial* Intelligence *(ICTAI)*, (pp. 625-631). doi:10.1109/ICTAI50040.2020.00101

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2014). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*. doi:10.18653/v1/N19-1423

Florea, A. M., Dascălu, M., Sîrbu, M. D., & Trăușan-Matu, Ș. (2020). Improving Writing for Romanian Language. *Project and Design Literacy as Cornerstones of Smart Education. Smart Innovation, Systems and Technologies. 158*, pp. 131-141. Singapore: Springer. doi:10.1007/978-981-13-9652-6_12

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017, July). Convolutional Sequence to Sequence Learning. *International Conference on Machine Learning* (pp. 1243-1252). PMLR. Retrieved June 21, 2020, from https://arxiv.org/abs/1705.03122

Graves, A. (2013). Generating Sequences With Recurrent Neural Networks. *arXiv preprint arXiv:1308.0850*. Retrieved June 20, 2020, from https://arxiv.org/abs/1308.0850

Iamandi, V. (2017). Romanian Spelling and Grammar Checking Systems. *Conference of Mathematical Society of the Republic of Moldova, 4*, pp. 509-514. Chișinău.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002, July). Bleu: a Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th annual meeting of the Association for Computational Linguistics* (pp. 311-318). Association for Computational Linguistics. doi:10.3115/1073083.1073135

Phi, M. (2020). *Illustrated Guide to Transformers: Step by Step Explanation.* Retrieved June 19, 2020, from Michael Phi: https://www.michaelphi.com/illustrated-guide-to-transformers

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. *arXiv preprint arXiv:1409.3215*. Retrieved June 20, 2020, from https://arxiv.org/abs/1409.3215

TensorFlow. (2019). *Transformer model for language understanding.* Retrieved June 10, 2020, from TensorFlow: https://www.tensorflow.org/tutorials/text/transformer

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention Is All You Need. *arXiv preprint arXiv:1706.03762*. Retrieved June 10, 2020, from https://arxiv.org/abs/1706.03762

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., . . . Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144*. Retrieved June 20, 2020, from https://arxiv.org/abs/1609.08144