Towards Lithuanian Grammatical Error Correction

Lukas Stankevičius[0000−0003−0012−5471] and
Mantas Lukoševičius[0000−0001−7963−285X]

Faculty of Informatics, Kaunas University of Technology, Kaunas, Lithuania

Abstract. Everyone wants to write beautiful and correct text, yet the lack of language skills, experience, or hasty typing can result in errors. By employing the recent advances in transformer architectures, we construct a grammatical error correction model for Lithuanian, the language rich in archaic features. We compare subword and byte-level approaches and share our best trained model, achieving $F_{0.5}$ = 0.92, and accompanying code, in an online open-source repository.

Keywords: Natural Language Processing · Grammatical Error Correction · Transformer Models · ByT5 · Lithuanian

1 Introduction

Recent advances in neural Natural Language Processing (NLP) have pushed the frontiers. In particular, transformer-architecture-based models [25] surpassed human performance in various NLP benchmarks such as SQuAD2.0 [16], GLUE [27], and SuperGLUE [26]. This also opened new opportunities for the Grammatical Error Correction (GEC) task which we address in this work.

GEC is the task of correcting different kinds of errors in text such as spelling, punctuation, grammatical, and word choice errors. The abundance of such noise in the text can hinder not only the understanding by humans but also the performance of various downstream NLP systems. An error-free text is also more beautiful, clean, associated with a certain prestige. However, producing it may be problematic for non-native speakers, language learners, it requires additional time and effort.

NLP state of the art for GEC still has much room to improve. As of now, the best $F_{0.5}$ scores are only up to 0.72[1]. Moreover, the research is mostly focused on English and a few other popular languages. To reduce this gap and contribute to the GEC progress, in this work, we investigate it for the Lithuanian language.

The Lithuanian language is one of the oldest living languages in the world. It has retained most of the features of the Indo-European Protolanguage, i.e., it is characterized by a very ancient linguistic structure: declensions (of nouns, adjectives, and pronouns), short and long vowels, diphthongs, etc. Lithuanian also has many similarities with Sanskrit – the classical language of ancient India,

[1] `http://nlpprogress.com/english/grammatical_error_correction.html`

still used today as a scholarly and liturgical language in Hinduism, Buddhism, and Jainism. Antoine Meillet (1886-1936), one of the most influential French linguists, once stated: "Anyone wishing to hear how Indo-Europeans spoke should come and listen to a Lithuanian peasant".

The Lithuanian language is synthetic and uses inflections to express syntactic relationships within a sentence. In other words, the relations in a sentence are expressed by word endings rather than with unbound morphemes and word order. This allows a lot of freedom in composing sentences. In contrast to agglutinative languages, which combine affixes by "gluing" them unchanged inside word ending, in Lithuanian inflectional categories are "fused". Meanwhile, prefixes, suffixes, and infixes are still used to derive words. Lithuanian verbs can be made from any onomatopoeia; phrasal verbs (e.g., go in, go out) are composed by adding the prefix to the verb. Lithuanian is unique for having 13 different participial forms of the verb [9] while modern English has only 2 (present and past participles). It is estimated that 47% of Lithuanian word forms are morphologically ambiguous [17], i.e., requiring context consideration to discern the meaning. All these mentioned features of the Lithuanian language make it interesting and important to analyze in the context of automatic GEC.

Our contributions are:

- We present the first GEC system for Lithuanian language based on deep neural networks.
- We compare sub-word and byte-level tokenization approaches for Lithuanian grammatical error correction.
- We share all the technical details, code, and model weights for open reuse and reproducibility.

2 Related Work

The simplest form of GEC is spellcheck. GNU Aspel[2] and Hunspell[3] are two widely-used open source spellcheckers. In particular, Hunspell [5] is the only system we found for Lithuanian GEC. Such systems work by keeping a large dictionary of possible words and detecting the non-words. During detection, the nearest alternatives from the dictionary are suggested. In Hunspell's case, the dictionary is made more compact by keeping only the main word forms with transformation rules, prefixes and suffixes. Spellchek systems are compact but limited to the correction of only non-words.

The first systems for a more complex GEC were based on Statistical Machine Translation (SMT) using a noisy channel model [2]. A significant contribution to GEC was the introduction of the CoNLL-2014 shared task [12]. Multiple systems were proposed, and among them, the phrase-based SMT setup was the most promising [8]. Yet neural approaches started to emerge, like [29]. As such systems advanced, hybrid statistical (SMT) and Neural Machine Translation

[2] `http://aspell.net/`

[3] `http://hunspell.github.io/`

(NMT) approaches [7] took the top. Only the introduction of the Transformer model [24] enabled neural approaches to supersede the statistical ones. As of now, the latter systems are claiming state-of-the-art results in GEC [13, 18].

Novel less-supervised approaches are also emerging. A simple language model reaching a reasonable performance with minimal annotated training data was demonstrated in [3]. The proposed system used n-gram language model to score variations of a sentence until incremental inflections do not improve the score anymore. Such a system was again improved using transformer-based language models instead of the n-grams in [1]. It turns out that such a less-supervised approach can outperform fully-supervised systems that were claiming state-of-the-art results several years ago.

Currently, the main constraint for GEC is the lack of training data. Researchers make progress by including new data sources or using automatic grammatical error generation to synthesize them. A simple language-agnostic pretraining objective was proposed in [18]. The authors automatically corrupted sentences in character level: swapping, inserting, dropping spans; token level: swapping, dropping spans; word level: lower-casing, upper-casing the first letter. A bigger model and larger dataset allowed achieving state-of-the-art GEC results for 4 languages. Authors of [19] used a small corpus of spelling errors to derive statistics for typographical error generation and generate a large parallel synthetic corpus. Another way is to use the data that the model incorrectly predicted during the training. A fluency boost learning and inference mechanism was proposed in [6] that reuses less fluent model predictions as new inputs during subsequent epochs. Similar trends are emerging with other languages. For example, simply adding new data improved a Transformer GEC system for the Czech language [11]. To summarize, it is important for neural GEC systems to be trained on large and high-quality corpora.

3 Dataset

As mentioned, a large dataset is essential for training a neural GEC solution. The data also has to be of the highest quality so that we can take it as a gold standard of grammatically-correct text. The largest publicly available general-purpose dataset for the Lithuanian language is from OSCAR [14] at 5 GB of deduplicated text. However, it is obtained from a general Common Crawl[4] and makes trusting the grammatical correctness problematic.

To make sure that the text is of good quality, we crawled various Lithuanian websites ourselves. We manually checked how the text is structured in every webpage so that only the relevant parts: title, summary (optionally), and the main text paragraphs would be scrapped. We crawled the following types of web pages: news, literature, blogs, encyclopedias, others.

We added titles and summaries to the main text paragraphs as additional paragraphs. Finally, we split the data into paragraphs. As a result, a single paragraph became a single sample of our dataset.

[4] `https://commoncrawl.org/`

3.1 Preprocessing

Although we performed a well-curated data scraping, there were still some artifacts in our data that had to be corrected or removed.

We had to remove some websites because of relatively high rates of spelling errors. This left us with a total of 34 final websites.

Some common error patterns can be easily corrected automatically. We looked at common mistakes in Lithuanian web texts [23] and performed the following corrections:

1. Incorrect quotation marks. In Lithuanian, the correct are „ABC“. Meanwhile, the English version “ABC” or others such as the universal "ABC" is often used instead.
2. The lack of space. It can happen before “m.” and “d.” abbreviations. For example, the text “1918m. vasario 16d.” must be corrected to “1918 m. vasario 16 d.”. Additionally, the space is often omitted after a full stop: “ir t.t.” and “A.Sabonis” must be corrected to “ir t. t.” and “A. Sabonis”.
3. An unnecessary space. The space must be omitted before most punctuation marks: “tik darbui , visiškai pamirštant poilsį ,” is corrected to “tik darbui, visiškai pamirštant poilsį,”.

We also filtered the text samples based on some statistics:

1. The sample text length should be at least 20 characters.
2. The fraction of Lithuanian letters in a sample should be at least 0.98. This filters out text from other languages and with miscellaneous characters. In the end, we are solving a task for the Lithuanian language. We included the characters “€£$%wx” as Lithuanian since they are used quite often.
3. The fraction of spaces to non-spaces should be at most 0.02. This allowed us to filter out samples dominated by URL addresses.

Lastly, we deduplicated our text samples. We shuffled the resulting 29 312 785 samples and took a subset of 4 194 304 for this work. Some statistics for the subset are depicted in Table 1.

Table 1. Dataset sizes by various tokenizations. The total dataset size is 4 194 304 samples.

Tokenizer	Sample length, mean ± std	Tokens, $\times 10^6$	Tokenization example
Characters	226 ± 194	947	Lietuva – graži šalis
ByT5 [30]	243 ± 194	1 017	Lietuva \xe2\x80\x93 gra\xc5\xbei \xc5\xa1alis
T5 [21]	48 ± 43	201	[_Lietuva] [_–] [_graži] [_šalis]
mT5 [31]	71 ± 61	298	[_] [Lietuva] [_–] [_] [graž] [i] [_šal] [is]
Words	30 ± 26	126	[Lietuva] [graži] [šalis]

The transformer model has a quadratic running time complexity $\mathcal{O}(n^2)$ with respect to the sequence length n (number of tokens). Usually, this is not a constraint as most text tasks are within the maximal sequence length of 512 (T5 [15]) sub-words or 1024 (ByT5 [30]) bytes. Yet in our training dataset, we had longer examples that we did not wish to truncate and, hence, lose. Instead, we split these too-long sequences to the length of 2100 characters for the T5 model and 700 characters for the ByT5. After that, we proceeded with the corresponding tokenization. As a result, the exact numbers of samples and tokens differ for both models, but the initial dataset and the amount of text (see Table 1) is the same.

For both runs, we set aside 0.05% of the data for validation and another 0.05% for testing.

4 Methods

4.1 Generating Grammatical Errors

We induce 3 groups of synthetic grammatical errors described below.

Typographical Errors They are induced by modeling the way how humans mistype on the keyboard. We follow the exact same methodology as in [22]: take mistyping statistics between each pair of characters on a QWERTY keyboard from an English dataset and apply them probabilistically to our texts. Out of the all characters considered, this way we corrupted 2% of them; from which were: substitution, 36.1%; deletion, 31.7%; insertion, 17.8%; transposition, 14.4%.

Confusing Similar Sounding Letters This is a very common source of spelling mistakes. We model them by defining sets of characters that sound alike, and randomly substituting a letter with one from the same set at a point of the generated error. For this, we use weighted sampling. The probabilities of the letters for substitution are proportional to how frequent they are in Lithuanian texts overall. For example, a in single set of letters "iįy" (where sounds differ only in their length), it is way more common to mistakenly write "i" instead of "į", rather than "į" instead of "i", as "i" is much more common. Only 2% of all such found occurrences were replaced. Groups of letters and detailed probabilities for the group members are derived from the raw (no preprocessing) subset of 2 909 403 samples, and are presented in Appendix A.

Other Errors We also introduce errors in the text by the four specific rules described below. We, again, thus corrupt 2% of the matches of the rules.

1. Gemination are doubled consonant letters that sound like a single one and thus is prone to be typed only once. This also applies to any consecutive letters from "cčsšzž" group. For example, the words "pusseserė užsimerkė" may be mistakenly written as "puseserė usimerkė" as they sound similar due to the gemination.

2. Assimilation to an adjacent letter. This is specific to any letter of "ptkssš" being before any of the "bdgzž" or vice versa. For example, the words "dirbti, lipdavo" may be mistakenly written as "dirpti, libdavo" as this is how they sound due to the assimilation.
3. Uppercasing or lowercasing the first letter in a word. For example, the word "ąžuolas" can start both with the lower or upper case depending on whether it is a tree (oak) or person's name. We exclude the first words in a sentence as these always have to start the upper case.
4. Delete and add space. We separately match all occurrences of spaces and all empty strings not at a word boundary.

Some samples of the corrupted sentences are presented in Appendix B.

4.2 Transformer Models

In this work, we compare T5 [21] and ByT5 [30] transformer models for grammatical error correction of Lithuanian. They are of sequence-to-sequence type. The encoder encodes the input sequence with attention operating on all input tokens while the decoder predicts output sequence tokens one by one, attending to tokens of both encoder (all) and decoder (only previous ones).

Below we further emphasize the properties of these models that make them appropriate for our task.

T5 The original T5 [15] was designed to be universal for multiple tasks. Authors showed that there is no difference whether a custom "head" is used (added on top of the pre-trained transformer) for fine-tuning purposes or a simple sequence-to-sequence formulation in text format is employed (no need to add additional weights to a pre-trained model). This way even tasks with outputs as float numbers can be formatted into a text-to-text format. Such generic task formulation made the T5 model very popular.

In previous work [21], we adapted the T5 model for the generation of summaries of Lithuanian news articles. We trained a SentencePiece [10] tokenizer on 10^6 and the main model on 2 027 418 news articles. As a result, this model should be familiar with the Lithuanian language (both tokenizer and model weights) and we use it as the basis for our fine-tuning purposes.

ByT5 ByT5 is a follow-up model from the multilingual mT5 [31] and T5 [15]. The authors showed that adapting byte-level tokenization can lead to a much more efficient use of model parameters. As an example, the multilingual mT5 had over 66% of its weights (for the base version) allocated to its multilingual word pieces (a total of 250 000) related weights (input embedding matrix and decoder softmax layer) which were only sparsely updating during the training. Meanwhile, ByT5 vocabulary has only 384 items and the model reuses the saved parameters in more massive layers rather than indexing tokens. These benefits allowed ByT5 to surpass the small and the base versions of mT5 [30].

The introduction of finer byte-level tokenization is especially important for grammatical error correction. Typos, variants in spelling and capitalization, and morphological changes can lead to completely different sub-word tokens, while byte tokens are affected the least. The authors of ByT5 showed that their model outperforms mT5 if various types of noise are introduced. Therefore, we use this model in our study of Lithuanian grammatical error correction.

4.3 Training Details

To train the models, we used a GeForce RTX 2080 Ti GPU. Following the best practices with the T5 family of models [15,18,30,31], we used the total batch size (number of samples to pass through the model before the gradient update) of 128 for fine-tuning. For ByT5 it was achieved by 128 gradient accumulation steps of batch size 1; while for T5, 64 gradient accumulation steps of batch size 2. We had to use multiple accumulation steps to process the total batch sequentially by smaller parts as the total batch did not fit into GPU memory at once. It took us approximately 100 hours for ByT5 and 30 hours for T5 fine-tuning. ByT5 took longer due to the longer sequences produced by finer byte-level tokenization.

We used the training script and Pytorch model implementation from the Hugging Face library [28]. For simplicity, we employed an Adafactor optimizer [20] with a constant learning rate of 0.001. If not stated otherwise, we used all the default parameters as in the Hugging Face library version 4.12.0.

4.4 Evaluation

One of the most popular grammatical error correction evaluation metrics is ERRANT [4]. It applies a set of rules operating over a set of linguistic annotations to construct the alignment and extract individual edits between corrupted, corrected, and gold-standard texts. This way precision, recall, and F-score can be calculated. We customized the original ERRANT by using Hunspell dictionaries [5], stemmer[5], spaCy version 3.2 pipeline `lt_core_news_lg` [6], and corresponding part-of-speech tags for the Lithuanian language.

During the inference, we used simple greedy decoding with a beam size of 1. That is, we simply selected for each next token the one that the model assigned the highest probability to.

5 Results

Training dynamics of T5 and ByT5 models are depicted in Figure 1. Only after 6% of training, the ByT5 score $F_{0.5} = 0.85$ is already higher than the $F_{0.5} = 0.80$ T5 managed to reach after the full epoch. We can also see that the performance is steadily increasing during the fine-tuning and is expected to continue doing

[5] `https://pypi.org/project/PyStemmer/`

[6] `https://spacy.io/models/lt`

so. The same results are indicated by the training loss. It is much lower for the ByT5, hence the model is better.

We divided our synthesized errors into several groups and corrupted the test set with each group separately from the others. Evaluation results of such setup are presented in Table 2. We can see that the easiest task for both models was adding or deleting spaces, while the hardest task is correcting assimilation and gemination mistakes. This group may lag in performance due to the smaller abundance (2% of samples) in the training data.

We present some generated test samples in Appendix B.

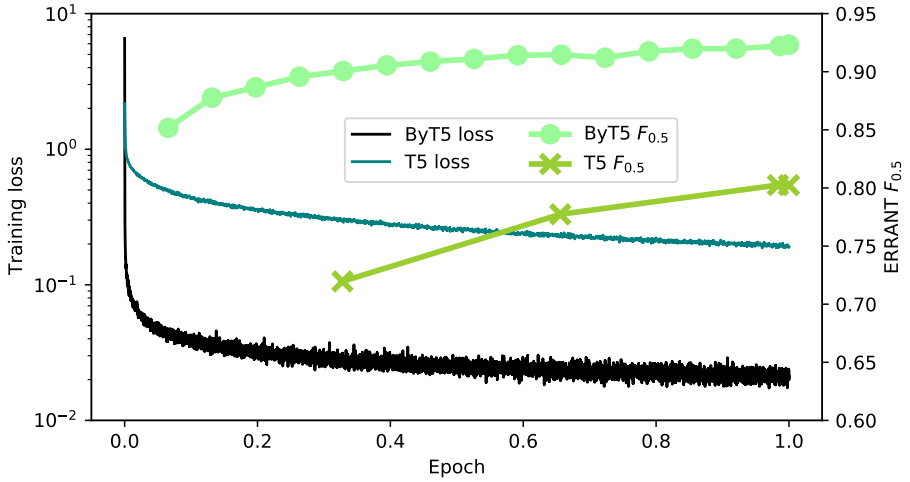

Fig. 1. Training loss and $F_{0.5}$ score for both T5 and ByT5 runs.

Table 2. Evaluation for the separate error categories with models trained for one epoch. We applied synthetic corruption for the test set of ByT5 (total of 2 155 samples) and T5 (total of 2 099 samples) with each error group separately. We show both ERRANT $F_{0.5}$ score and number of samples (#samples) affected and evaluated on.

Error group	ByT5		T5	
	$F_{0.5}$	#samples	$F_{0.5}$	#samples
Typographical	0.87	1 916	0.72	1 868
Punctuation	0.81	489	0.36	460
Similar sounding letters	0.88	1 115	0.55	1 143
Add/delete spaces	0.96	1 873	0.74	1 832
Assimilation/Gemination	0.79	56	0.30	43
Upper/Lower casing	0.86	785	0.47	781

6 Discussion

We trained the first reported deep-learning-based Lithuanian grammatical error correction system and compared two sequence-to-sequence transformer models for the task.

The ByT5 transformer model, based on byte-level tokenization, greatly outperformed the subword counterpart T5. We think that the main reason for this is that the fine-grained byte-level details allow the model to maximize acquired information about the sentence and thus calculate a more accurate representation. This way, the model sees a bigger picture and has to solve the task with less ambiguity. On the other hand, longer and more informative token sequences are slower to process and induce the slowdown of three times, compared to the T5. Yet even if we compare models trained for the same amount of time, ByT5 is still the leader. This shows that for the grammatical error correction it is crucial to have the best possible representation of the text.

We thought that during the T5 subword tokenizer training acquired common token patterns may be of great use. Yet our results show that this is not the case. On the contrary, it may make it harder for the model to "understand" the true representation behind the corrupted text.

In the future, we plan to train the ByT5 model even longer. It is clearly visible from our results that in the current state it is under-trained. Additional benefits could be expected from more data and more passes through the dataset.

We hope that this work will help both researchers and Lithuanian language users. We make our trained model and code available at `https://github.com/LukasStankevicius/Towards-Lithuanian-Grammatical-Error-Correction`.

Funding

The research is partially funded by the joint Kaunas University of Technology Research and Innovation Fund and Vytautas Magnus University project "Deep-Learning-Based Automatic Lithuanian Text Editor (Lituanistas)", Project no.: PP34/2108.

Acknowledgements

We thank our project collaborators from Vytautas Magnus University, especially Jurgita Kapočiūtė-Dzikienė, for valuable discussions on related topics.

References

1. Alikaniotis, D., Raheja, V.: The unreasonable effectiveness of transformer language models in grammatical error correction. In: Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications. pp. 127–133. Association for Computational Linguistics, Florence, Italy (Aug 2019). https://doi.org/10.18653/v1/W19-4412, `https://aclanthology.org/W19-4412`

2. Brockett, C., Dolan, W.B., Gamon, M.: Correcting ESL errors using phrasal SMT techniques. In: Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics. pp. 249–256. Association for Computational Linguistics, Sydney, Australia (Jul 2006). https://doi.org/10.3115/1220175.1220207, https://aclanthology.org/P06-1032
3. Bryant, C., Briscoe, T.: Language model based grammatical error correction without annotated training data. In: Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications. pp. 247–253. Association for Computational Linguistics, New Orleans, Louisiana (Jun 2018). https://doi.org/10.18653/v1/W18-0529, https://aclanthology.org/W18-0529
4. Bryant, C., Felice, M., Briscoe, T.: Automatic annotation and evaluation of error types for grammatical error correction. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 793–805. Association for Computational Linguistics, Vancouver, Canada (Jul 2017). https://doi.org/10.18653/v1/P17-1074, https://aclanthology.org/P17-1074
5. Dadurkevičius, V.: Assessment data of the dictionary of modern lithuanian versus joint corpora (2020), http://hdl.handle.net/20.500.11821/36, CLARIN-LT digital library in the Republic of Lithuania
6. Ge, T., Wei, F., Zhou, M.: Reaching human-level performance in automatic grammatical error correction: An empirical study. arXiv preprint arXiv:1807.01270 (2018)
7. Grundkiewicz, R., Junczys-Dowmunt, M.: Near human-level performance in grammatical error correction with hybrid machine translation. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). pp. 284–290. Association for Computational Linguistics, New Orleans, Louisiana (Jun 2018). https://doi.org/10.18653/v1/N18-2046, https://aclanthology.org/N18-2046
8. Junczys-Dowmunt, M., Grundkiewicz, R.: Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. pp. 1546–1556. Association for Computational Linguistics, Austin, Texas (Nov 2016). https://doi.org/10.18653/v1/D16-1161, https://aclanthology.org/D16-1161
9. Klimas, A.: Some unique features of Lithuanian. Lituanus **30**(3), 51–64 (1984)
10. Kudo, T., Richardson, J.: SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 66–71 (2018)
11. Náplava, J., Straka, M., Straková, J., Rosen, A.: Czech grammar error correction with a large and diverse corpus. arXiv preprint arXiv:2201.05590 (2022)
12. Ng, H.T., Wu, S.M., Briscoe, T., Hadiwinoto, C., Susanto, R.H., Bryant, C.: The CoNLL-2014 shared task on grammatical error correction. In: Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task. pp. 1–14. Association for Computational Linguistics, Baltimore, Maryland (Jun 2014). https://doi.org/10.3115/v1/W14-1701, https://aclanthology.org/W14-1701
13. Omelianchuk, K., Atrasevych, V., Chernodub, A., Skurzhanskyi, O.: GECToR – grammatical error correction: Tag, not rewrite. In: Proceedings of the Fif-

teenth Workshop on Innovative Use of NLP for Building Educational Applications. pp. 163–170. Association for Computational Linguistics, Seattle, WA, USA → Online (Jul 2020). https://doi.org/10.18653/v1/2020.bea-1.16, https://aclanthology.org/2020.bea-1.16
14. Ortiz Suárez, P.J., Sagot, B., Romary, L.: Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures. In: Bański, P., Barbaresi, A., Biber, H., Breiteneder, E., Clematide, S., Kupietz, M., Lüngen, H., Iliadi, C. (eds.) 7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7). Cardiff, United Kingdom (7 2019). https://doi.org/10.14618/IDS-PUB-9021, https://hal.inria.fr/hal-02148693
15. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research **21**(140), 1–67 (2020), http://jmlr.org/papers/v21/20-074.html
16. Rajpurkar, P., Jia, R., Liang, P.: Know what you don't know: Unanswerable questions for SQuAD. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 784–789. Association for Computational Linguistics, Melbourne, Australia (Jul 2018). https://doi.org/10.18653/v1/P18-2124, https://aclanthology.org/P18-2124
17. Rimkutė, E.: Morfologinio daugiareikšmiškumo ribojimas kompiuteriniame tekstyne [Morphological Disambiguation of the Corpus of Lithuanian Language]. Ph.D. thesis, Vytautas Magnus University, Kaunas (2006), english summary available at https://etalpykla.lituanistikadb.lt/object/LT-LDB-0001:E.02~2006~1367155963435/E.02~2006~1367155963435.pdf
18. Rothe, S., Mallinson, J., Malmi, E., Krause, S., Severyn, A.: A Simple Recipe for Multilingual Grammatical Error Correction. In: Proc. of ACL-IJCNLP (2021)
19. Shah, K., de Melo, G.: Correcting the autocorrect: Context-aware typographical error correction via training data augmentation. In: Proceedings of the 12th Language Resources and Evaluation Conference. pp. 6930–6936. European Language Resources Association, Marseille, France (May 2020), https://aclanthology.org/2020.lrec-1.856
20. Shazeer, N., Stern, M.: Adafactor: Adaptive learning rates with sublinear memory cost. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 4596–4604. PMLR (10–15 Jul 2018), https://proceedings.mlr.press/v80/shazeer18a.html
21. Stankevičius, L., Lukoševičius, M.: Generating abstractive summaries of lithuanian news articles using a transformer model. In: International Conference on Information and Software Technologies. pp. 341–352. Springer (2021)
22. Stankevičius, L., Lukoševičius, M., Kapočiūtė-Dzikienė, J., Briedienė, M., Krilavičius, T.: Correcting diacritics and typos with byt5 transformer model. arXiv preprint arXiv:2201.13242 (2022)
23. Tamulionienė, A., et al.: Būdingiausios rašybos klaidos mokinių rašiniuose ir tinklaraščiuose. Bendrinė kalba (iki 2014 metų–Kalbos kultūra) (88), 1–24 (2015)
24. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

25. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 6000–6010. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (2017)
26. Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.: Superglue: A stickier benchmark for general-purpose language understanding systems. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019), https://proceedings.neurips.cc/paper/2019/file/4496bf24afe7fab6f046bf4923da8de6-Paper.pdf
27. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.: GLUE: A multi-task benchmark and analysis platform for natural language understanding. In: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. pp. 353–355. Association for Computational Linguistics, Brussels, Belgium (Nov 2018). https://doi.org/10.18653/v1/W18-5446, https://aclanthology.org/W18-5446
28. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., Rush, A.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 38–45. Association for Computational Linguistics, Online (10 2020). https://doi.org/10.18653/v1/2020.emnlp-demos.6, https://aclanthology.org/2020.emnlp-demos.6
29. Xie, Z., Avati, A., Arivazhagan, N., Jurafsky, D., Ng, A.Y.: Neural language correction with character-based attention. arXiv preprint arXiv:1603.09727 (2016)
30. Xue, L., Barua, A., Constant, N., Al-Rfou, R., Narang, S., Kale, M., Roberts, A., Raffel, C.: Byt5: Towards a token-free future with pre-trained byte-to-byte models (2021)
31. Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., Raffel, C.: mT5: A massively multilingual pre-trained text-to-text transformer. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 483–498. Association for Computational Linguistics, Online (6 2021). https://doi.org/10.18653/v1/2021.naacl-main.41, https://aclanthology.org/2021.naacl-main.41

A Statistics for corrupting similar letters and punctuation

Table 3. Regex expressions to find specific patterns in texts and statistics of distinct finds used as probability weights for replacement.

Group (regular expression)	Matches and counts					
`[,\.-]{0,1}␣`	␣	79 695 056	.␣	5 125 941		
	,␣	9 876 726	-␣	1 347 515		
`[\.,;:\-\-?!\(\)\[\]\<\>/]`	,	10 072 919	?	300 962	]	34 283
	.	7 976 435	:	519 928	>	5 759
	-	1 453 095	!	106 333	<	4 457
	)	665 253	;	105 526		
	(	655 651	/	90 778		
	–	546 698	[	34 295		
`u{0,1}ou{0,1}`	o	33 058 916	ou	41 509		
	uo	3 355 463	uou	34		
`ia|e`	ia	6 733 731	e	35 509 427		
`[scz]`	s	47 349 069	c	2 645 328	z	1 646 823
`[ščž]`	š	7 002 598	č	2 619 317	ž	5 044 500
`[eęė]`	e	35 509 427	ę	1 336 170	ė	9 781 460
`[iįy]`	į	3 490 952	y	8 347 510	i	82 431 807
`[uųū]`	ū	2 795 974	ų	7 826 828	u	28 978 236
`[aą]`	a	68 291 558	ą	4 471 872		
`[cč]`	c	2 645 328	č	2 619 317		
`[zž]`	z	1 646 823	ž	5 044 500		
`[td]`	t	35 864 854	d	14 822 144		
`[kg]`	k	26 461 947	g	10 626 341		
`[pb]`	p	16 187 509	b	8 148 725		
`“|,,|[,,“””]|”`	"	436 378	,,	11 777	”	87
	”	46 847	“	817		

B Corruption and correction examples

Table 4. Samples of the original, corrupted, and corrected text forms. Here fully-trained (1 epoch) ByT5 models were used.

Type	Text
Original	„Mes nenorime, kad jie keiktųsi, pyktųsi. Neleidžiame ne tik gerti, bet ir rūkyti. Taisyklės čia griežtos, rūkei, atleisime tau kartą, nepaklusai, eik iš kur atėjęs. Jei jau žmogus nusprendė keisti gyvenimą, tai turi būti daroma rimtai“, - nuolaidų nežada M. Balčiūnas.
Corrupted	"Mes nenorime, kad jie keiktųsi, byktųsi. Nleeidžiame ne tik gerti, bet ir rūkyti. Taisyklės čia griežtos, Rūkei, atle isime tau kartą, nepaklusai, eik iš kur atėjęs. Jei j au žmogus nuspr-endė keisti gyvenimą tai turi būti daromo rymtai“, - nuolaidų nežada M. Balčiūnas]
ByT5	„Mes nenorime, kad jie keiktųsi, pyktųsi. Neleidžiame ne tik gerti, bet ir rūkyti. Taisyklės čia griežtos. Rūkei, atleisime tau kartą, nepaklusni, eik iš kur atėjęs. Jei jau žmogus nusprendė keisti gyvenimą, tai turi būti daroma rimtai“, - nuolaidų nežada M. Balčiūnas.
Original	Šeštadienio vakarą Klaipėdoje surengto „Eurovizijos“ atrankos finalo dalyviai po renginio miegoti nėjo – dešimt savaičių trukusios kovos pabaigą atšventė uostamiesčio kokteilių bare „Oscar“.
Corrupted	Šeštdienio vakarą Klaipėdoje surengto „Eurovizijos“ atrankos finalo dalyvia i po rengicio miegoti nėjo – dešimt savaičių drukusio kovos pabaigą atšventė uostamiesčio kokteilių bare „oscar“.
ByT5	Šeštadienio vakarą Klaipėdoje surengto „Eurovizijos“ atrankos finalo dalyviai po renginio miegotinėjo – dešimt savaičių trukusio kovos pabaigą atšventė uostamiesčio kokteilių bare „Roscar“.
Original	300 kg hašišo gabenimo į Lietuvą byla: vienas išteisintas, kitam sušvelninta bausmė
Corrupted	300 kg haši šo gabenimo į Lietuvą byla. vie nas i štsisintas, kitam sušverlninta buasmė
T5	300 kg hašišo gabenimo į Lietuvą byla: vienas išteisintas, kitam sušvelninta bausmė