ISSN 2007-9737

No Need to Get Wasteful: The Way to Train a Lightweight Competitive Spelling Checker Using (Concentrated) Synthetic Datasets

Vladimir Starchenko*

Higher School of Economics University, School of Linguistics, Moscow, Russia

vmstarchenko@edu.hse.ru

Abstract. This study focuses on spelling checkers, which remains problematic for modern error correction systems. Based on T5 architecture, we create a lightweight spelling check tool that can be used in combination with a large language model (LLM) and significantly improves the overall result of the error correction system. It also performs competitively compared to other recently developed spelling check tools, despite being considerably smaller in size. The high performance of the model is obtained as a result of introducing two synthetic datasets: a dataset with a high density of spelling errors and the dataset with errors more difficult for correction.

Keywords. Spelling errors, spelling check, grammatical error correction, preprocessing, synthetic datasets.

1 Introduction

The performance of large language models (LLMs) on the task of error correction in natural texts has greatly improved over recent years. [26] shows that T5 and GECToR receive as high an evaluation in standard metrics as human experts do, while subsequent studies, e. g., [40, 41], further improve their results.

Still, particular kinds of errors or errors in particular environments remain difficult for language models to correct. [26] notices errors related to inter-sentence dependencies, errors in long sentences, errors in certain syntactic configurations, etc.

Crucially, one kind of errors that large and accurate modern GEC models handle notoriously unsatisfactorily are character-level errors or, simply put, spelling errors. Spelling errors are easily corrected by humans and are rated simplest in studies that compare the difficulty of errors in texts [9].

Yet since the start of active use of machine learning in GEC and until now, researchers pervasively notice that SOTA GEC systems perform poorly with spelling errors [6, 32], among others.

[32] note that GEC systems may not only fail to correct a single spelling error, but also get triggered to hallucinate both in the word with an error or elsewhere in the sentence. [36] suggests similar effects for the interaction of spelling errors with other NLP tasks like summarization and question-answer systems.

As a solution to this problem, we suggest a light-weight system which is specifically dedicated to correcting spelling errors in English and can be used jointly with a large GEC model. We train and evaluate models of various architectures and sizes and select 31M T5-efficient-mini [36] as demonstrating the best recourse consumption–performance ratio. The model is publicly available[1]. We suggest a training algorithm consisting of several steps.

[1] huggingface.co/startc/optispell-t5-mini

Table 1. CSD: Example

Correct sentence	This guarantees secure transmission and is extremely useful to companies sending/receiving critical information.
Corrupted sentence	T**t**his guaranee**n** seecur**et**ransmi**es**o**o**ion and is extr**ee**ly u**u**seful to comp**y**an**es** s**d**endng/rec**ie**ving critical in**fr**mation.

It includes training on synthetic data that imitates the errors met in natural texts, tuning on generated dataset with selected more difficult errors[2] and on natural texts (dataset clang-8; [28]). The model outperforms SOTA spelling correcting systems, cf. [21], and consume considerably less resources than them (e.g., 59.55 MB versus 1.37 GB).

The rest of the paper is organized as follows: section 2 overviews related studies. Section 3 suggests the definition of spelling errors in the present study. Section 4 discusses the synthetic and natural datasets. Section 5 presents the experiments, evaluating the influence of model size, architecture and training process on the performance and comparing the best result with other recently developed spelling check tools. Section 6 provides the discussion of the experiments. The last section is the conclusion.

2 Related Work

Spelling checking systems for English have been developed since the earliest stages of electronic text editing tools. Multiple approaches were suggested, including rule-based approaches [15, 35, 37], among others, statistical approaches [1, 14, 33, 38], among others, and machine learning [3, 21, 24], among others. The correction of spelling errors is commonly assumed a part of GEC task: a GEC system is expected to output a correct text akin to one that an educated speaker would produce. Despite this fact, as noted, GEC models struggle when dealing with spelling [32, 6, 29, 34, 40]. Spelling check is performed by some GEC systems for various languages as a preprocessing step [4, 6, 7, 30].

This step is however not considered standard (e.g., it is not mentioned in the recent overview of GEC systems [2]), and no standard tool for this task exists. One reason that prevents the usage of the tools listed above for this purpose is the commonly accepted wide notion of spelling, including within this category many error types.

[24], developing a common benchmark for spelling correction tools, discusses errors in verbal tense, pronoun choice and punctuation among 12 suggested types of spelling errors. Confusion of semantically distant words is also usually included in spelling errors, e.g., [21, 24], among many others.

This approach to the definition of a spelling error is problematic in the way that it does not draw a clear line between spelling, on the one hand, and grammar, word choice and other types of errors, on the other hand. This division is however crucial for approaches using LLMs. As it has been discussed, they show poorer results with character-level errors (spelling in the narrow sense) but show outstanding results with token-level errors (other types), prompting to treat the former separately.

3 Definition of a Spelling Error

The present study uses the following definition for a spelling error:

(1) A spelling error is a distortion in the written text, which creates a non-existent word or word form. Further, we refer to the errors defined by (1) as spelling errors in the narrow sense when contrasted to the wider and more traditional understanding of this notion.

The definition (1) comprises two crucial components. Firstly, it suggests that spelling errors are defined at the level of word forms, meaning that for locating them, no access to the context is needed. Greatly simplifying, one could build a system that finds spelling errors in the narrow

[2] Both synthetic datasets are available by the link: huggingface.co/datasets/startc/synthetic-spelling.

Table 2. NCSD: Example

Correct sentence	Fedora Linux replaced the original Red Hat Linux download and retail version.
Corrupted sentence	Fedora Linux replaced the original Red Hat Linux download and **ee**tail version.

Table 3. Optimal dataset sizes

Dataset	Size
CSD	16M
NCSD	8M
clang-8	2.34M

sense based purely on a dictionary of existing word forms. Access to the context is however essential for correction of errors, so that the result corresponds to the word meant by the author and semantically fits the text.

Secondly, the definition (1) excludes various cases that are often mentioned among spelling errors, but lead to the creation of existing words. Such cases include words that are clearly distinct semantically, but are graphically or phonetically close. Possible examples include *way* instead of *away*, *form* instead of *from*.

The framework of the present study suggests that such errors that go beyond spelling in the narrow sense are dealt with by a large GEC model. These models solve the tasks related to grammar or semantics, including word choice, with a high accuracy, independently of whether the erroneous option looks similar to the correct one or it does not.

The overall performance of a spelling checker and a LLM is not expected to improve after additional tuning of a spelling checker to such errors. It is thus the spelling in the narrow sense that must be handled by a spelling checker.

The errors that create non-existent words may correspond to various errors in more fine-grained classifications, e.g., [16, 25]. In some cases, incorrect spelling arises from typing errors: incidental letter deletion, insertion, replacement or transposition.

Errors of this kind are distributed more randomly than the other ones and correlate with the positions of keyboard buttons. In other cases, spelling errors may arise in orthographically difficult contexts, e.g., *kik* instead of *kick* in the texts of children or language learners, *hypertention* instead of *hypertension* for more proficient speakers.

Unlike the previous group, orthographic errors are distributed less randomly and associated with letters or digraphs that are easy to confuse or particular lexemes that are difficult for speakers or language learners to write.

Lastly, a misspelling may result from an ill-formed word formation process. These include both inflection, e.g., non-existent forms like *gooses* or *medias*, and derivation, e.g., *disclosement* instead of *disclosure*.

The understanding of the sources of spelling errors is important for the framework that uses synthetic datasets for training: the distribution of various kinds in the synthesized data must correspond to the actual distribution or include enough data of each type for relevant patterns to be learnt by the LLM.

4 Datasets

4.1 Training Set

In the training workflow we use multiple training datasets. For pre-training we create a large synthetic dataset with a high density of spelling errors. Fine-tuning is performed on several smaller tuning synthetic datasets and clang-8, comprising natural language data.

We suggest a novel approach to the generation of synthetic data, as the existing approaches to the generation of spelling errors use a different error definition [3, 20, 21, 24], etc., see section 3.

Concentrated synthetic dataset (CSD)[3]**.** The optimal suggested pipeline (see subsection 5.3 for more detail) includes the usage of synthetic data with a high density of spelling errors. The generation of this dataset is as follows:

[3] Available by the link: huggingface.co/datasets/startc/synthetic-spelling

Table 4. Performance of T5-efficient model, various sizes

Model	Prec	Rec	$F_0.5$	Model Size, MB	Model Size, parameters
T5-base	0.925	**0.773**	0.890	425.15 MB	222M
T5-efficient-small	0.930	0.770	0.892	115.41 MB	60M
T5-efficient-mini	**0.936**	0.768	**0.897**	59.55 MB	31M
T5-efficient-tiny	0.912	0.711	0.864	29.7 MB	15M

1. As a source of correct sentences, we used the dataset of Wikipedia articles[4]. The whole dataset was separated into segments of up to 63 tokens after tokenization (on average, about 63 × 4 letters and 1–4 sentences). This grouping is optimal for GPU working memory and does not affect the quality of the model after further tuning.
2. In each fragment, 0 to 30 words are randomly selected for corruption. The number of corrupted words is around 15–20 with a lower frequency of a smaller and larger numbers.
3. The selected words are subjected to one of corrupting transformations:
   (a) Deletion of a letter,
   (b) Adding a letter in the word,
   (c) Doubling a letter,
   (d) Swap of adjacent letters,
   (e) Replacement of a letter with another one,
   (f) Deletion of one of double letters,
   (g) Adding a letter which is close on the keyboard to one of adjacent letters,
   (h) Replacement of a letter into another one which is close on the keyboard.

   Notice that some of the transformations may lead to the same results. For example, corrupted words produced by the deletion of one of double letters are a subset of results produced by the deletion of a random letter ($f \subset a$); same is true for $g \subset b$ and $h \subset e$.

   Despite this fact, transformations f–h are included as a separate option for the corruption algorithm in order to raise the frequency of errors that a human is more likely to make.
4. Errors that humans tend to make in natural texts, including errors in usage of similar letters or digraphs and word formation, were added to the words from a predefined list[5].
5. Errors related to space placement were added: a random space may be inserted inside a word or deleted.

Interjections, proper names and words shorter than 4 characters were not selected for corruption. If the error generating algorithm created a sequence of letters corresponding to an existing word, it was rerun.

The algorithm of corrupted text generation allows us to expose a model to various types of errors described in the previous section. Steps 3 and 5 add typographical errors, while step 4 provides orthographically difficult contexts and erroneous word formation.

Table 1 shows an example from the CSD dataset. It includes the original correct sentence and its corrupted version with 12 errors of various types, induced according to the algorithm described above.

Non-concentrated synthetic dataset (NCSD)[6]. The first step of fine-tuning was made on a separate synthetic dataset. The aim of using this dataset is two-fold.

[4] Available by the link: huggingface.co/datasets/legacy-datasets/wikipedia

[5] The list of frequent errors in English words is based on their long list at the Wikipedia, available by the link: en.wikipedia.org/wiki/Wikipedia:List_of_spelling_variants

[6] Available by the link: huggingface.co/datasets/startc/synthetic-spelling

Table 5. Performance of various versions of T5 and BART

Architecture	Prec	Rec	$F_{0.5}$
T5-efficient-small	0.923	0.768	**0.887**
T5-small	0.922	0.759	0.884
long-T5	**0.933**	0.720	0.881
BART-base	0.889	**0.786**	0.866
byT5	0.879	0.698	0.836

First, it exposes the model to less erroneous data to lower the number of false positive results. Secondly, it provides more complicated examples. To achieve the latter, examples with 1–2 erroneous words per sentence were generated and checked by a model trained without fine-tuning by NCSD (only CSD and clang-8 used).

The examples that this model fails to process correctly were added to the dataset, mixed with examples that are completely correct. Table 2 presents an example from the NCSD dataset with one of two possible erroneous words.

Clang-8. As the second step of fine-tuning we use the dataset clang-8. It is created from the NAIST Lang-8 Learner Corpora which comprises natural texts of L2 English speakers and is further cleaned by a SOTA GEC system [28]. All the errors except for spelling were removed from the dataset.

4.2 Dataset Sizes

The size of training datasets is crucial for the resulting performance of a model. It is often true that the larger a dataset, the more information a model learns. On the other hand, with the growth of the training dataset, the pace of model enhancement decreases and resource consumption grows.

The present setup potentially allows the scaling of the synthetic datasets infinitely, so it is important to recognize the point until which it is expedient to enlarge them. Optimal suggested dataset sizes are listed in Table 3; sizes for CSD and NCSD are given in segments, as described in the beginning of this section, for Clang-8 — in sentences. In order to find them, we assess the models from subsection 5.2 trained on synthetic datasets of sizes until the point the performance (almost) stops growing (see subsection 5.3 for more detail). Clang-8 consists of natural erroneous texts and can only be scaled by (half)-manual annotation, which is beyond the limits of the present study. Given its relatively small size, we take it in its entirety.

We notice that testing the models trained on different dataset sizes reveals that the pace of training does not necessarily correlate with the best possible result.

Model A may perform significantly worse than model B trained on a small synthetic dataset ($\leq$1M), yet demonstrate an advantage when trained on larger dataset.

4.3 Evaluation Set

For evaluation, we use the JFLEG dataset [23]. We calculate Precision, Recall, and $F_{0.5}$ scores, which is widely used for the GEC task and represents human judgments in this task well [5, 10, 22].

This makes our results comparable with recent studies dedicated to spelling. We focus on comparison with [21] as the best available results.

Our definition of spelling errors section 3 makes the evaluation only partly comparable. Some errors, e.g., confusion of phonologically similar existing words or grammar errors that do not lead to creation of non-existent words, are not considered spelling errors within our approach, but are supposed to be corrected in the other systems.

JFLEG comprises a fraction of such errors annotated as spelling errors. As a result, we create a modification of JFLEG which does not have existing (but incorrect) words annotated as spelling errors, henceforth JFLEG-NE.

We use JFLEG-NE for the evaluation of our models and provide the scores for both versions of JFLEG when comparing our models with the results of other studies.

Table 6. Training workflows

Training steps	Prec	Rec	$F_0.5$
CSD (16M) + NCSD + clang-8	**0.927**	**0.756**	**0.887**
CSD (24M) + clang-8	0.921	0.743	0.879
CSD (16M) + clang-8	0.918	0.746	0.878
CSD (16M) + NCSD	0.903	0.741	0.865
CSD (24M)	0.918	0.658	0.851
CSD (16M)	0.913	0.660	0.848

5 Experiments

5.1 Experiment 1: Model Size

One of the aims of the present study is creating a high-performance spelling checker that can be paired with a large GEC model without significant increase in the resource consumption of the whole system. In order to make a spelling checker lightweight, we test different sizes of the same model. The scores are presented in Table 4.

Table 4 shows that the size of the model has a small effect on its performance on the spelling correction task. Even the "tiny" model shows good results, with a difference in decimals of $F_{0.5}$ score. With the growth of model size over $31M$ parameters $F_{0.5}$ score even slightly decreases: the rise of recall with the model size is compensated with the fall of precision.

5.2 Experiment 2: Model Architecture

Recent studies mark T5 [27] as the best performing model architecture for the spelling check task [21]. We test various types of T5 in addition to the plain T5. Efficient-T5 [36] has a different model shape, which performs better in certain tasks. Long-T5 [11] is suggested to show better results for longer sentences.

ByT5 [39] does not exploit a tokenization mechanism and is expected to perform better on character-level tasks like spelling check. As a comparison, we take another architecture: BART [18], which was originally trained to reconstruct corrupted text and is claimed to perform well on the GEC task in general [13].

Notice that we do not balance the models with respect to their size. The previous section shows that from some point the model size does not significantly affect performance.

Table 5 shows the performance of the models. Expectedly, various versions of T5 show comparable results with the best result for T5-efficient-small.

5.3 Experiment 3: Training Workflow

In this section we focus on setting a workflow for training a spelling checker with the best performance. First, we discuss the optimal combination of training datasets. Second, we comment on the hyperparameters of the models.

5.3.1 Datasets Used

Table 6 presents the evaluation for various workflows for training a spelling check model based on the combinations of datasets presented in subsection 4.1 (for model T5-efficient-mini).

The combination of all three training datasets shows the best result from the point of view of $F_{0.5}$, Precision and Recall. Training on synthetic CSD with simpler errors yields the lowest score, while adding either NCSD with more difficult errors or Clang-8 with natural data leads to comparable enhancement.

As noted, at some point, increasing the synthetic dataset no longer noticeably improves the performance of the system, compare in particular "CSD (16M) + clang-8" and "CSD (24M) + clang-8".

5.3.2 Training Hyperparameters

The performance of a model mostly depends on the architecture and training datasets. However, parameters of training may affect the result (up to $\Delta F_{0.5} = 0.02$) or the convergence speed.

The optimal learning rate depends on the model architecture and size and was separately found for every combination ($8 \cdot 10^{-3}$ on CSD / NCSD, $8 \cdot 10^{-4}$ on clang-8 for T5-efficient-mini as the best-performing model). Similarly, optimal effective batch size affects the performance (96 for T5-efficient-mini).

Table 7. Comparison of best solutions of the present study and by [21]

Dataset	Model	Prec	Rec	$F_0.5$
JFLEG	T5-efficient-mini	**0.936**	0.768	**0.897**
JFLEG	T5-large	0.906	**0.855**	0.895
JFLEG-NE	T5-efficient-mini	**0.934**	0.933	**0.934**
JFLEG-NE	T5-large	0.833	**0.957**	0.855

The spelling check model allows for the use of bf16 [12] without loss of quality, resulting in the reduction by half of the model size (in MB) and training time.

5.4 Experiment 4: Comparison with Previous Studies

[21] evaluates several recent and best performing models for the spelling check task. We compare our solution with the best model in terms of F-score, according to their results.

Performance. The performance of the two models is only partially comparable due to the different definition of a spelling error. The sets of errors which are supposed to be corrected by our framework and [21] do not coincide.

[21] corrects errors that lead to creation of existing words (the wide definition of a spelling error); we correct all cases that lead to creation of non-existent words, while [21] distinguishes cases in which a non-dictionary form is not an error, including dialect forms, intentional word corruptions and the like. Provided that such cases are considerably rare, a close approximation is that the errors that spelling errors in the narrow sense that we correct are a subset of errors corrected by models in [21].

In order to make the results more comparable, we evaluate models over two versions of JFLEG. One is its original version (with all errors except for spelling corrected), which corresponds to the definition of a spelling error by T5-large [21] and therefore is expected to show better performance of this model. JFLEG-NE only includes errors that lead to emergence of non-existent words and is expected to give an advantage to our system. Noticeably, T5-efficient-mini shows a slightly higher $F_{0.5}$ score for both evaluation datasets, despite JFLEG includes errors that are not supposed to be corrected by this model. This effect is achieved as a result of a high precision, while the recall is higher for T5-large [21]. For the JFLEG-NE, the difference grows even further: recall of T5-efficient-mini improves with the elimination of errors that create existing words, and resulting $F_{0.5}$ score delta reaches $\Delta F_{0.5} = 0.079$.

Resource consumption. As Table 9 shows, T5-efficient-mini features 31M parameters, whereas T5-large is more than 20 times larger, featuring 737M parameters. It yields a difference in the resource consumption: the smaller model is lighter or faster in various respects, including weight, execution speed and training costs (the latter is also influenced by different training procedures). The detailed comparison is presented in the Table 8 (59.55 MB 1.37 GB).

Makes both its weight () and performance time insignificant compared to those of a large GEC model (cf., and for the model by [21].

6 Discussion

In this section, we discuss the four experiments presented above in turn. We conclude that the goal of creating a lightweight and competitive spelling checker is achieved and elaborate on the methods that allow to reach this result, on their further application and restrictions.

Experiment 1: Model size. An important result of this experiment is that the spelling check task does not require a large model size. Enlarging the model over 31M parameters does not yield any performance improvement.

Recall of larger models slightly grows, yet results in more hallucination and lower precision, as well as great expenses in learning and

Table 8. Comparison of T5-efficient-mini and T5-large [21]

Property	T5-efficient-mini	T5-large
Size, parameters	31M	737M
Weight	59.55MB	1.37 GB
Performance speed, CPU	104.4 examples per second (JFLEG)	5.277 examples per second (JFLEG)
Performance speed, GPU	2843 examples per second (JFLEG) with the optimal batch size 4096	154.4 examples per second (JFLEG) with the optimal batch size 128; with the growth of the batch size the speed decreases
Training time	9 hours using one GeForce RTX3090 (24GB memory)	360 hours on eight Nvidia A100 GPUs

application time and memory. This result may follow from the fact that in most cases the correction of a spelling error does not require appealing to the semantics, which in its turn sets high requirements to the model size.

The smallest best-performing model has the weight of 59.55 MB, which is about a tenth or lower of a regular large GEC model. Further reduction of a model is impractical, as it leads to a lower quality.

Experiment 2: Model architecture. T5 architecture confirms its high performance in the spelling check task. Different versions of this architecture do not greatly vary with the highest score belonging to T5-efficient.

BART architecture shows high performance with respect to recall, but loses in $\Delta F_{0.5}$ due to its low precision. These facts suggest that this model in general tends to induce more changes into a text, whether it is erroneous or correct.

Experiment 3: Training workflow. The suggested training workflow makes use of two large synthetic datasets. The idea to rely on synthetic data when solving the spelling check task is natural, provided that the erroneous spelling is relatively easy to synthesize, and many previous studies exploit it [8, 19, 21, 24], among others. The two innovations that were introduced concern the structure of the synthetic data used. The pre-training CSD dataset is concentrated: it includes 15–20 erroneous words per 1–4 sentences on average. Such a high proportion of erroneous words allows to expose a model to the wider range of corrupted segments. It is particularly fruitful for the spelling check task, as spelling errors create a great space of variation and are problematic as they require generalization beyond patterns seen in the training data [4].

Further tuning on the data with lower error density prevents a model from the excessively high rate of false positives, caused by its “habit” to meet many errors in one sentence. The second innovation is introducing a dataset which includes errors which are particularly difficult for a model (NCSD). The approach using this dataset suggests a two-step training process.

The first step is creating a model in the absence of this dataset and probing synthetic data for the errors which this model fails to correct; the second iteration makes use of these difficult errors. Adding NCSD into the training workflow raises the performance of a model compared to using both only CSD ($\Delta F_{0.5} = 0.017$) and CSD + clang-8 ($\Delta F_{0.5} = 0.009$).

Noticeably, the combination of only synthetic datasets (CSD + NCSD) with no exposure to the natural data yields results that are comparable with the best performing full system. It means that the suggested workflow (concentrated synthetic dataset + synthetic dataset with difficult errors) can be successfully used for languages with no available natural corpus.

Experiment 4: Comparison with previous studies. The suggested combination of model architecture, size and training workflow create a SOTA spelling check model with a low resource consumption. Firstly, it slightly outperforms the

Table 9. A large GEC model with and without preprocessing by our spelling checker

Models	Prec	Rec	$F_0.5$
T5-efficient-mini (spelling) + BART-large (GEC)	**0.7382**	**0.5702**	**0.6971**
BART-large (GEC)	0.7314	0.502	0.6702

SOTA model suggested by [21] both for the wide definition and to a greater degree to the narrow definition of a spelling error.

Secondly, the trained model features only 31M parameters, which makes both its weight (59.55MB) and performance time insignificant compared to those of a large GEC model (cf., 737M parameters and 1.37 GB for the model by [21].

Applying the developed spelling check tool as a preprocessing stage prior to a large GEC model improves the result of the whole system. To exemplify it, we choose BART additionally trained for the GEC task [13]. Table 9 shows the performance of BART on its own and in conjunction with the spelling checker.

One has to notice that the JFLEG dataset contains a relatively low proportion of spelling errors. A possible explanation for it is that JFLEG consists of English learner texts for the TOEFL exam.

The situation of examination encourages students to pay a particular attention to the well-formedness of texts, and the spelling errors are expected to be easier to notice for non-native speakers after double-checking an examination paper.

A similar effect is observed for other evaluation datasets, see [32] for the discussion of ConLL-2014, which comprises essay texts [25]. In contrast, native speakers writing everyday texts pay less attention to checking their texts, yet in general write grammatically correctly. As a result, the proportion of spelling errors in such texts is expected to be higher, yielding their greater influence on the overall performance of a GEC system.

7 Conclusions

In this study, we created a lightweight spelling check system based on T5 architecture, whose function is to eliminate non-existent words. This model can be applied as a preprocessing step in GEC systems with large language models, enhancing the overall result on the GEC task. Our model shows the best result in this task, compared to existing spelling checking systems.

In addition, the created model reaches SOTA results in correcting spelling errors in a wide sense. A small size of our spelling check tool allows using it on its own on the computing systems with few resources.

The high performance of a model is achieved by using two synthetic datasets, one featuring high density of errors, the other comprising selected more difficult errors. Provided that both datasets are automatically generated from well-formed natural sentences, this approach may be scaled to other languages that do not have a corpus of erroneous texts.

The question that mostly stays unaddressed in this study is the reason for the poor performance of large GEC models on spelling errors. This problem is even more striking provided that the spelling errors are particularly easy to correct for a human.

While we make a step towards the practical solution of this problem by suggesting the use of a specialized tool for the spelling check task, one could claim that a more desired solution is to cure this flaw for large GEC models. One possible explanation for this disadvantage of GEC models compared to human experts lies in common tokenization algorithms.

People see words in alphabetic languages as sequences of characters, and a spelling correction usually concerns an operation with one (rarer more than one) of characters within a sequence. In contrast, language models that are used for the GEC task commonly tokenize a text at the preprocessing stage. It allows to treat separately frequent segments that are associated with a specific semantics (affixes and frequent roots) and at the same time have a limited size for a model dictionary with no unattested words (see e.g., WordPiece tokenization used in T5 [17, 31]). As a

result, what appears to require a slight alteration in terms of characters corresponds to a great change in terms of tokens. This difference is further aggravated by the fact that spelling errors often disrupt regular morphemes and an erroneous word splits into more tokens than a regular word of the same size.

As an example, consider a false correction made by the T5 model trained in this study: misspelled *spouce* in a particular example it corrects into *source* instead of *spouse*. The character sequence *spouce* is split by SentencePiece tokenizer [17] into 4 items. The erroneous correction thus presents from the perspective of a model as *spouce* [3, 7990, 76, 565] $\rightarrow$ *source* [1391], while the right correction is *spouce* [3, 7990, 76, 565] $\rightarrow$ *spouse* [9911]. Although for a human observer *spouse* is clearly closer to *spouce* than *source*, for the model both are a replacement of four tokens into one.

Thus, tokenization, which greatly improves processing the semantics of a text, obscures spelling. Testing architectures, our study tests a straight-forward solution for the tokenization problem, training byT5 among other architectures. It operates on byte-level and is not expected to face the difficulties described above. Nevertheless, byT5 does not perform on the spelling check task better than the T5 with a regular tokenizer (Table 5), suggesting that a more elaborate approach is needed. We leave finding such an approach for further research.

References

1. **Ahmed, F., Luca, E. W. D., Nürnberger, A. (2009).** Revised n-gram based automatic spelling correction tool to improve retrieval effectiveness. Polibits, Vol. 1, No. 40, pp. 39–48.
2. **Bryant, C., Yuan, Z., Qorib, M. R., Cao, H., Ng, H. T., Briscoe, T. (2023).** Grammatical error correction: A survey of the state of the art. Computational Linguistics, Vol. 49, No. 9, pp. 643–701. DOI: 10.1162/coli_a_00478.
3. **Büyük, O., Arslan, L. M. (2021).** Learning from mistakes: Improving spelling correction performance with automatic generation of realistic misspellings. Expert Systems, Vol. 38, No. 5, pp. e12692. DOI: 10.1111/exsy.12692.
4. **Chollampatt, S., Ng, H. T. (2017).** Connecting the dots: Towards human-level grammatical error correction. Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications, Association for Computational Linguistics, pp. 327–333. DOI: 10.18653/v1/W17-5037.
5. **Chollampatt, S., Ng, H. T. (2018).** A reassessment of reference-based grammatical error correction metrics. Proceedings of the 27th International Conference on Computational Linguistics, pp. 2730–2741.
6. **Chollampatt, S., Wang, W., Ng, H. T. (2019).** Cross-sentence grammatical error correction. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 435–445. DOI: 10.18653/v1/P19-1042.
7. **Ge, T., Wei, F., Zhou, M. (2018).** Fluency boost learning and inference for neural grammatical error correction. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, Vol. 1, pp. 1055–1065. DOI: 10.18653/v1/P18-1097.
8. **Ghosh, S., Kristensson, P. O. (2017).** Neural networks for text correction and completion in keyboard decoding. arXiv. DOI: 10.48550/arXiv.1709.06429.
9. **Gotou, T., Nagata, R., Mita, M., Hanawa, K. (2020).** Taking the correction difficulty into account in grammatical error correction evaluation. Proceedings of the 28th International Conference on Computational Linguistics, International Committee on Computational Linguistics, pp. 2085–2095. DOI: 10.18653/v1/2020.coling-main.188.
10. **Grundkiewicz, R., Junczys-Dowmunt, M., Gillian, E. (2015).** Human evaluation of grammatical error correction systems. Proceedings of the 2015 Conference

on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, pp. 461–470. DOI: 10.18653/v1/D15-1052.

11. **Guo, M., Ainslie, J., Uthus, D., Ontanon, S., Ni, J., Sung, Y. H., Yang, Y. (2022).** LongT5: Efficient text-to-text transformer for long sequences. Findings of the Association for Computational Linguistics: NAACL 2022, Association for Computational Linguistics, pp. 724–736. DOI: 10.18653/v1/2022.findings-naacl.55.
12. **Kalamkar, D., Mudigere, D., Mellempudi, N., Das, D., Banerjee, K., Avancha, S., Teja-Vooturi, D., Jammalamadaka, N., Huang, J., Yuen, H., Yang, J., Park, J., Heinecke, A., Georganas, E., Srinivasan, S., Kundu, A., Smelyanskiy, M., Kaul, B., Dubey, P. (2019).** A study of BFLOAT16 for deep learning training. arXiv. DOI: 10.48550/arXiv.1905.12322.
13. **Katsumata, S., Komachi, M. (2020).** Stronger baselines for grammatical error correction using a pretrained encoder-decoder model. Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pp. 827–832. DOI: 10.5715/jnlp.28.276.
14. **Kernighan, M. D., Church, K. W., Gale, W. A. (1990).** A spelling correction program based on a noisy channel model. COLING: Papers presented to the 13th International Conference on Computational Linguistics, Vol. 2, pp. 205–210.
15. **Kondrak, G., Sherif, T. (2006).** Evaluation of several phonetic similarity algorithms on the task of cognate identification. Proceedings of the Workshop on Linguistic Distances, pp. 43–50.
16. **Korre, K., Pavlopoulos, J. (2020).** ERRANT: Assessing and improving grammatical error type classification. Proceedings of the 4th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature, Online, pp. 85–89.
17. **Kudo, T., Richardson, J. (2018).** SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 66–71. DOI: 10.18653/v1/D18-2012.
18. **Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L. (2020).** BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703.
19. **Li, H., Wang, Y., Liu, X., Sheng, Z., Wei, S. (2018).** Spelling error correction using a nested RNN model and pseudo training data. arXiv. DOI: 10.48550/arXiv.1811.00238.
20. **Martynov, N., Baushenko, M., Abramov, A., Fenogenova, A. (2023).** Augmentation methods for spelling corruptions. Proceedings of the International Conference Dialogue.
21. **Martynov, N., Baushenko, M., Kozlova, A., Kolomeytseva, K., Abramov, A., Fenogenova, A. (2023).** A methodology for generative spelling correction via natural spelling errors emulation across multiple domains and languages. Findings of the Association for Computational Linguistics: EACL 2024, pp. 138–155. DOI: 10.48550/arXiv.2308.09435.
22. **Napoles, C., Sakaguchi, K., Post, M., Tetreault, J. (2015).** Ground truth for grammatical error correction metrics. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pp. 588–593. DOI: 10.3115/v1/P15-2097.

23. **Napoles, C., Sakaguchi, K., Tetreault, J. (2017).** JFLEG: A fluency corpus and benchmark for grammatical error correction. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Vol. 2, pp. 229–234. DOI: 10.48550/arXiv.1702.04066.

24. **Näther, M. (2020).** An in-depth comparison of 14 spelling correction tools on a common benchmark. Proceedings of the Twelfth Language Resources and Evaluation Conference, pp. 1849–1857.

25. **Ng, H. T., Wu, S. M., Briscoe, T., Hadiwinoto, C., Susanto, R. H., Bryant, C. (2014).** The CoNLL-2014 shared task on grammatical error correction. Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task, pp. 1–14. DOI: 10.3115/v1/W14-1701.

26. **Qorib, M. R., Ng, H. T. (2022).** Grammatical error correction: Are we there yet? Proceedings of the 29th International Conference on Computational Linguistics, pp. 2794–2800.

27. **Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J. (2020).** Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research, Vol. 21, No. 140, pp. 1–67.

28. **Rothe, S., Mallinson, J., Malmi, E., Krause, S., Severyn, A. (2021).** A simple recipe for multilingual grammatical error correction. Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pp. 702–707. DOI: 10.18653/v1/2021.acl-short.89.

29. **Rozovskaya, A., Roth, D. (2016).** Grammatical error correction: Machine translation and classifiers. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 2205–2215. DOI: 10.18653/v1/P16-1208.

30. **Sakaguchi, K., Post, M., van-Durme, B. (2017).** Grammatical error correction with neural reinforcement learning. Proceedings of the Eighth International Joint Conference on Natural Language Processing, pp. 366–372.

31. **Sennrich, R., Haddow, B., Birch, A. (2016).** Neural machine translation of rare words with subword units. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pp. 1715–1725. DOI: 10.18653/v1/P16-1162.

32. **Starchenko, V. M., Starchenko, A. M. (2023).** Here we go again: Modern gec models need help with spelling. Proceedings of the Institute for System Programming of the RAS, Vol. 35, No. 5, pp. 215–228. DOI: 10.15514/ISPRAS-2023-35(5)-14.

33. **Stüker, S., Fay, J., Berkling, K. (2011).** Towards context-dependent phonetic spelling error correction in children's freely composed text for diagnostic and pedagogical purposes. INTERSPEECH, pp. 1601–1604.

34. **Susanto, R. H., Phandi, P., Ng, H. T. (2014).** System combination for grammatical error correction. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 951–962. DOI: 10.3115/v1/D14-1102.

35. **Taghva, K., Stofsky, E. (2001).** OCRSpell: an interactive spelling correction system for OCR errors in text. International Journal on Document Analysis and Recognition, Vol. 3, No. 3, pp. 125–137. DOI: 10.1007/PL00013558.

36. **Tay, Y., Dehghani, M., Rao, J., Fedus, W., Abnar, S., Chung, H. W., Narang, S., Yogatama, D., Vaswani, A., Metzler, D. (2021).** Scale efficiently: Insights from pre-training and fine-tuning transformers. arXiv. DOI: 10.48550/arXiv.2109.10686.

37. **van-Delden, S., Bracewell, D., Gomez, F. (2004).** Supervised and unsupervised automatic spelling correction algorithms. Proceedings of the 2004 IEEE International Conference on Information

Reuse and Integration, pp. 530–535. DOI: 10.1109/IRI.2004.1431515.

38. **Vilares, J., Alonso, M. A., Doval, Y., Vilares, M. (2016).** Studying the effect and treatment of misspelled queries in cross-language information retrieval. Information Processing & Management, Vol. 52, No. 4, pp. 646–657. DOI: 10.1016/j.ipm.2015.12.010.
39. **Xue, L., Barua, A., Constant, N., Al-Rfou, R., Narang, S., Kale, M., Roberts, A., Raffel, C. (2022).** ByT5: Towards a token-free future with pre-trained byte-to-byte models. Transactions of the Association for Computational Linguistics, Vol. 10, pp. 291–306. DOI: 10.1162/tacl_a_00461.
40. **Zhang, Y., Zhang, B., Li, Z., Bao, Z., Li, C., Zhang, M. (2022).** SynGEC: Syntax-enhanced grammatical error correction with a tailored GEC-oriented parser. Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pp. 2518–2531. DOI: 10.18653/v1/2022.emnlp-main.162.
41. **Zhou, H., Liu, Y., Li, Z., Zhang, M., Zhang, B., Li, C., Zhang, J., Huang, F. (2023).** Improving Seq2Seq grammatical error correction via decoding interventions. Findings of the Association for Computational Linguistics: EMNLP 2023, pp. 7393–7405. DOI: 10.18653/v1/2023.findings-emnlp.495.

Article received on 09/07/2024; accepted on 17/10/2024.
**Corresponding author is Vladimir Starchenko.*