UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Martin Vainikko

Estonian Synthetic Error Generation by Prompting for Grammatical Error Correction

Master's Thesis (30 ECTS)

Supervisor(s): Agnes Luhtaru, MSc
Mark Fišel, PhD

Tartu 2024

Estonian Synthetic Error Generation by Prompting for Grammatical Error Correction

Abstract: For Estonian grammatical error correction (GEC), sufficient data to train end-to-end models is lacking. However, the recent advancements in large language models (LLMs) offer new opportunities. We utilize OpenAI's GPT models (GPT-3.5-Turbo, GPT-4-Turbo, and GPT-4) to generate synthetic errors and analyze these errors across different model versions, prompting strategies, and data domains. By fine-tuning models on these synthetic datasets and conducting human evaluations, we assess the effectiveness of various prompting strategies for synthetic error generation. Our findings indicate that within the GEC domain, the errors generated by GPT models are comparable to those made by humans. Human evaluations also revealed that GPT models produce problematic edits. This highlights significant potential for further research in this area.

Keywords:
grammatical error correction, GEC, low-resource, synthetic data, GPT-4, GPT-3.5, NLLB, large language model, LLM

CERCS: P176 Artificial intelligence

Suurte keelemudelite viipamine sünteetiliste grammatikavigade genereerimiseks eesti keeles

Lühikokkuvõte: Eesti keele grammatiliste vigade parandamise jaoks ei ole piisavalt andmeid, et tõhusalt treenida autokorrektorit. Hiljutised edusammud suurte keelemudelite vallas on aga avanud uusi võimalusi sünteetiliste andmete genereerimiseks. Genereerime OpenAI GPT mudelitega (GPT-3.5-Turbo, GPT-4-Turbo ja GPT-4) lausetesse grammatilisi vigu. Hindame genereeritud andmehulki manuaalselt hulki märgendades ning treenides hulkade peal transformeripõhiseid autokorrektoreid. Me järeldasime, et laused, mis pärinevad grammatiliste vigade korpusest ning kuhu GPT on vigu genereerinud, saavutavad automaathindmisel võrdväärseid tulemusi inimandmetega. Märgendamise tulemusena märkasime, et GPT genereerib probleemseid muutusi lausetesse. Töö avab palju uusi suundi edasisteks uurimusteks.

Võtmesõnad:
grammatiliste vigade parandamine, vähesed ressursid, sünteetilised andmed, GPT-4, GPT-3.5, NLLB, suur keelemudel

CERCS: P176 Tehisintellekt

Contents

1 Introduction

Grammatical error correction (GEC) involves automatically correcting the grammatical mistakes in an input text. A GEC tool, or autocorrection tool, identifies and corrects errors in text, which may include spelling, word order, word form, word choice, capitalization, and punctuation. Another aim of such tools is to enhance text fluency by eliminating officialese and simplifying complex sentences, thereby speeding up text editing. A well-known autocorrection tool for English is Grammarly[1].

Language is inherently complex. Linguistics describes language by morphology, syntax and semantics. The Estonian language's multidimensionality and numerous exceptions make developing a fully rule-based autocorrection model impractical. While rule-based approaches have their merits, more sophisticated methods are necessary to address the full spectrum of linguistic errors. Natural language processing (NLP) has advanced from statistical techniques such as bag of words to neural networks to transformer architectures. Recent developments in next-token prediction in auto-regressive language models, also known as large language models (LLMs), have shown substantial progress as few-shot[2] learners (Brown et al., 2020).

Similar trend has been taking place in GEC with transformer architecture being the current baseline. The advancements of large language models (LLMs) and their ability to generate grammatically correct text (Coyne et al., 2023) have posed new possibilities for GEC. One of the main bottlenecks for training or fine-tuning an end-to-end GEC model is the scarcity of GEC data, which holds for the Estonian language.

In this thesis, we evaluate the applicability of generating synthetic grammatical errors by prompting an LLM. To this extent, we take on the challenge of Estonian GEC data scarcity. We prompt different model versions of OpenAI's GPT[3] to generate errors to input sentences. We employ various prompting techniques such as zero-shot[4] and few-shot prompting, focusing on different few-shot prompting example sampling methods. Our main research questions are:

1. Is prompting GPT a viable technique for generating human-like synthetic grammatical errors?
2. How do different model versions affect error generation?
3. How do different prompting methods affect error generation?
4. How does the domain of source sentences affect error generation?

[1] `https://www.grammarly.com/`

[2] A prompting technique with examples of the task in the prompt, "few" indicating that there are multiple examples.

[3] `https://platform.openai.com/docs/models`

[4] A prompting technique without any examples of the task in the prompt.

To answer these questions, we perform a set of comparisons yielding 15 sets of parallel synthetic GEC data and 15 fine-tuned transformer-based MT models. We apply two methods for evaluation that involve comparing the yielded datasets to one another and a human GEC dataset:

- Fine-tuning an NMT model on each dataset and evaluating the results on a development set;
- Annotating a subset of sentences from a subset of datasets for a quantitative comparison.

We then present and discuss the results, acknowledging the limitations and future work of error generation by prompting.

The findings of this Master's thesis serve as a preliminary study for an experiment aimed at generating 100,000 synthetic error sentences in Estonian, Ukrainian, and German, as detailed in a preprint by Luhtaru et al. (2024b). The authors investigate various methods for generating synthetic errors, achieving state-of-the-art results for all three languages by fine-tuning a Llama model on synthetic data produced through back-translation for each language (Luhtaru et al., 2024b).

This thesis is organized as follows: in Chapter 2, we talk about the motivation and background of the work; Chapter 3 is dedicated to giving an overview of the experimental setup; in Chapter 4, the results of the experiments are given. This is followed by a discussion of the results, future work and conclusions.

OpenAI's ChatGPT (model version GPT-4) LLM and Grammarly were used for text editing. ChatGPT was prompted to correct the grammatical errors and make the text more fluent and clear. ChatGPT was also used to assist in generating LaTeX tables.

2 Background

In this chapter, we describe the background and terms related to the thesis. We provide a concise overview of the history of written Estonian and its grammar. We also explore transformers and large language models. Subsequently, we introduce grammatical error correction, discussing the various types of GEC models and how they are evaluated. We offer insights into Estonian GEC models, related work in synthetic grammatical error generation and additional background for the experimental setup.

2.1 Written Estonian

The following paragraph draws on the work of Erelt et al. (2020). Estonian is part of the Finnic branch of the Finno-Ugric languages and is spoken by approximately 1.1 million people, with 930,000 of these speakers residing in Estonia. It is believed that the influence of Baltic and Germanic languages merged the tribal dialects of Proto-Finnic ancestral language in the first half of the second millennium, leading to the emergence of the Estonian language. Estonian consists of two main dialect groups: Northern and Southern. The earliest records of the written Northern and Southern dialects date back to the 17th century. The translation of the Bible into the Northern dialect in 1739 significantly influenced the development of written Estonian, which is why it is predominantly based on the Northern dialect. The term "written language" or "written Estonian" (*kirjakeel*) refers to both the standard language and the language as it is physically written in Estonian. The standard language is a normalized language governed by a set of rules accepted nationwide. In Estonia, the Institute of the Estonian Language (*Eesti keele instituut* – EKI)[5] oversees the standard Estonian. EKI both preserves and updates the standard Estonian rules, vocabulary and suggestions. Language is not static and changes over time. The standard language is always a step behind the actual use of language.

The Estonian language is described by 14 cases and 38 inflection types (Erelt et al., 2018). An inflection type refers to words that exhibit similar word formation patterns. While the typical word order in Estonian is generally considered subject-verb-object (SVO), this is an oversimplification, as there are constructions that employ different word orders (Erelt, 2005). Moreover, the word order in Estonian is primarily influenced by the information structure of the sentence rather than the word's or phrases' role as a given clause element (Erelt et al., 2020). The extensive number of cases and inflection types contribute to the complexity of Estonian morphology. Additionally, the language's relatively flexible word order further complicates the evaluation of automatic grammatical error correction.

[5] `https://portaal.eki.ee/`

2.2 Transformers and LLMs

The following paragraph is based on the article by Vaswani et al. (2017). The transformer architecture is based on the neural network's architecture with core changes to how the hidden states are calculated. The authors introduced self-attention to a revised encoder-decoder NMT model architecture's multi-head attention layer. Their encoder (see Figure1) layer comprises two sub-layers: a multi-head attention and a feed-forward layer. There is a residual connection between the sub-layers followed by normalization. The decoder layer has a similar sub-layer structure with the addition of another multi-head attention sub-layer resulting in three sub-layers. The purpose of the additional multi-head attention layer is to perform attention over the output of the encoder stack. Self-attention minimizes the number of sequential operations by the attendance between a given position in the sequence and every other position in the sequence. The transformer structure allows parallel computations with less computational complexity by largely eliminating the sequential manner of calculating the hidden states like in a recurrent neural network (RNN) or a convolutional neural network (CNN). In addition to that, the architecture makes long-range dependencies in the network easier to learn thanks to shorter path lengths compared to previous NN architectures.

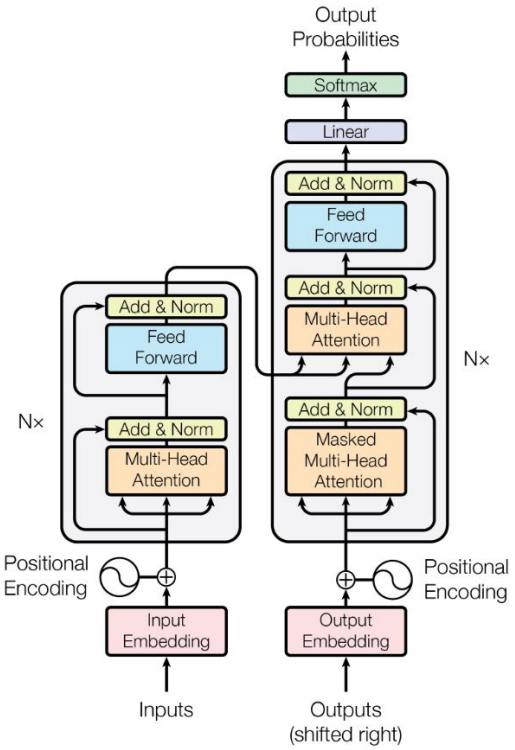

Figure 1. The transformer model architecture (Vaswani et al., 2017).

The transformer architecture is now a standard model in machine translation and large language models. An early example of a large language model using self-attention is the encoder-only BERT model, introduced by Devlin et al. (2019). This was followed by the decoder-only (see Figure 2) generative pre-training model (GPT-1) by Radford et al. (2018), which demonstrated that unsupervised pre-training, followed by task-specific fine-tuning, could achieve state-of-the-art results across multiple tasks. Radford et al. (2019) later introduced GPT-2, an improved and larger version of GPT-1 with 1.5 billion parameters, pre-trained on a vast web text corpus and achieving state-of-the-art performance in many tasks in a zero-shot setting, without downstream fine-tuning. GPT-2 was succeeded by GPT-3, a 175 billion parameter model trained on about two orders of magnitude larger dataset compared to GPT-2, achieving strong performance in zero- to few-shot setting (Brown et al., 2020). Subsequently, InstructGPT was introduced as a variant of GPT-3 aligned to follow the user's intent through reinforcement learning with human feedback (Ouyang et al., 2022). Despite having 100 times fewer parameters than GPT-3, InstructGPT was favored in human evaluations for its output (Ouyang et al., 2022). In 2023, GPT-4 was introduced, which was aligned post-training to desired behavior adherence (OpenAI, 2023).

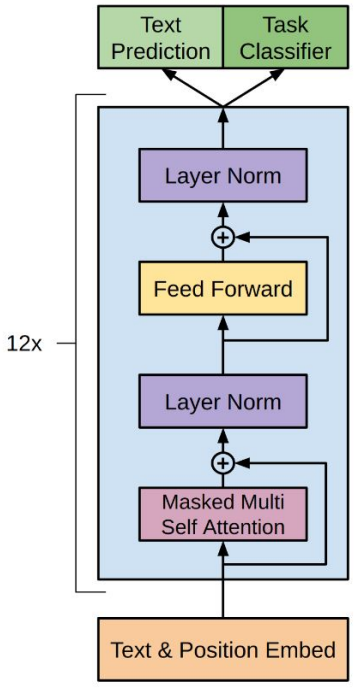

Figure 2. The architecture of GPT-1 (Radford et al., 2018).

In this work, we prompt different versions of GPT-3.5 and GPT-4 models aligned for instruction following.

GPT models have been trained on vast amounts of web text (Brown et al., 2020). Web texts can contain grammatical errors. Nevertheless, GPT models have learned to generate grammatically correct output as GPT models have shown good performance in GEC benchmarks (Coyne et al., 2023). A broader question the authors raise is whether GPT "remembers" the grammatical errors it has seen during training. Standard written language possesses a set of grammatical rules. There are no specific rules for producing grammatical errors, making any particular grammatical mistake statistically less likely than its correct form.

2.3 Grammatical Error Correction

Grammatical error correction tool or autocorrector takes text as input and returns the correct text correcting any grammatical errors. Similarly, a speller finds errors in text, but is limited to the word level. Early successful GEC systems were based on phrase-based statistical machine translation (SMT) methods by Brockett et al. (2006). Initially, neural methods in GEC did not achieve state-of-the-art results compared to phrase-based SMT baselines (Bryant et al., 2023). The emergence of neural machine translation (NMT) architectures, namely transformers, caused a paradigm shift in GEC by still being the leading architecture for GEC systems (Bryant et al., 2023). For Estonian, there is a rule-based speller developed by Filosoft[6]. Currently, the state-of-the-art GEC model for Estonian is a Llama-2-based model trained on back-translation style learned synthetic data + human data (Luhtaru et al., 2024b).

One of our synthetic data evaluation strategies involves training an machine translation model for GEC. We follow the approach by Junczys-Dowmunt et al. (2018) but fine-tune a multilingual MT model to be efficient and cope with the lack of training data. We fine-tune NLLB-200 for GEC, a transformer model by (Team et al., 2022). NLLB-200 (No Language Left Behind) is trained for machine translation between 200 languages (Team et al., 2022). The model is open-source, and one of the 200 languages it was trained on is Estonian, making it a valid choice for fine-tuning for Estonian GEC.

GEC systems are most often evaluated with a scorer that compares system output with human annotations/corrected text. MaxMatch (M^2), a scoring tool for GEC evaluation, was introduced by Dahlmeier and Ng (2012). The authors improved upon the shortcomings of the Helping Our Own (HOO) (Dale and Kilgarriff, 2011) evaluation script, which failed to recognize some correct edits. They developed an algorithm that identifies phrase-level edits and measures the maximum overlap with the gold standard, calculating the F_1 score, which averages precision and recall.

For our experiments, we utilize a version of the MaxMatch scorer that has been adapted for Estonian[7]. This modified scorer is designed to detect edits within word

[6] https://www.filosoft.ee/html_speller_et/

[7] https://github.com/TartuNLP/estgec/tree/main/M2_scorer_est

order and calculates the $F_{0.5}$-score, emphasizing precision more heavily than recall. The decision to use $F_{0.5}$-score is from (Ng et al., 2014). Other evaluation strategies in GEC literature include ERRANT (Bryant et al., 2017) and GLEU (Napoles et al., 2017).

2.4 More Technical Information

Our experiments include comparing different example sampling techniques in few-shot prompting.

One of the sampling methods is sampling by the closest Levenshtein distance (Levenshtein, 1966), which is an edit distance algorithm between strings calculated by the number of character deletions and insertions.

Another sampling technique is sampling by the closest cosine similarity between Sonar (Duquenne et al., 2023) embeddings. Sonar embedding space features an encoder-decoder model initialized with NLLB 1B dense model for encoding and decoding sentence-level embeddings (Duquenne et al., 2023).

2.5 Synthetic Error Generation

There are various methods for generating synthetic GEC errors. Grundkiewicz et al. (2019) extracted confusion sets from a spellchecker and utilized it for introducing errors to monolingual data, whose method has also been applied to Estonian alongside other languages by Luhtaru et al. (2024a). Zhao et al. (2019) applied a copying mechanism that copies unchanged words in the source sentence to the target sentence. In addition to probabilistic methods, Luhtaru (2022) trained a transformer model for GEC via synthetic pre-training. Luhtaru et al. (2024b) investigated various methods for generating synthetic errors, achieving state-of-the-art results for Estonian, Ukrainian and German by fine-tuning a Llama[8] model on synthetic data produced through back-translation for each language.

[8] https://llama.meta.com/

3 Experimental Setup

The main goal of the thesis's experiments is to compare synthetic grammatical error generation with GPT to human grammatical errors. We compare synthetic errors directly to human errors by generating synthetic errors in the source sentences of a GEC corpus, namely the UT L2 train set. To this extent, we explore using different versions of the GPT model, different prompting strategies, and source data from outside the GEC domain.

The experimental setup is structured into two main parts: 1) utilizing GPT models to introduce errors into input sentences using various prompting techniques, and 2) assessing the "quality" or human-likeness of the errors produced by these methods. The first part explores different prompting strategies with different domain sentences including zero-shot and few-shot prompting, alongside diverse methods for selecting examples in few-shot prompting. The second part is divided into two subsections: a) qualitative and quantitative analysis of the errors generated, and b) fine-tuning sequence-to-sequence models with the generated sentences, followed by a comparison of precision, recall, and F0.5 scores against human data and between different methods.

Additionally, this chapter provides an overview of the currently available Estonian GEC data and the M^2 scorer that was modified for Estonian.

3.1 The Plan

We evaluate synthetic error generation with GPT by changing different factors for generating errors:

1. GPT version;
2. Prompting method;
3. Example choosing method for few-shot prompting;
4. Seed data.

We introduce synthetic errors directly to UT L2 learner's corpus source sentences. This allows us to compare syntetic grammatical errors to grammatical errors made by humans. We also include another set of sentences from Estonian National Corpus for error generation source sentences. These sentences are not related to GEC data. For comparability, we match the number of sentences to UT L2 learner's corpus. In total, we generate 15 sets of synthetic GEC data.

3.2 Error Generation

3.2.1 Data

Estonian GEC data consists of two test sets, a development and a train set (see Table 1 for sizes). University of Tartu learner's (UT-L2) Train[9] consists of sentences written by students learning Estonian on B1 and B2 level (Rummo and Praakli, 2017). Estonian L2 Grammatical Error Correction Corpus (EstGEC-L2) Dev and Test sets[10] consist of sentences of essay-like writings and formal and informal letters on A2, B1, B2 and C1 levels (Luhtaru et al., 2024c). EstGEC-L2 Dev and Test sets have been tagged to M^2 format by human annotators.

Set	Nr. of Sentences
EstGEC-L2 Dev	1,761
EstGEC-L2 Test	2,029
UT-L2 Train	8,921
ENC-GEC	8,921

Table 1. Estonian GEC data and a subset of ENC corpus for error generation with thr number of sentences.

We also extract a subset from Estonian National Corpus 2021 (ENC) (Koppel and Kallas, 2022) for error generation. We sample an equal number of random sentences from ENC's Fiction, Web and Wikipedia subsections and match the length of UT-L2 (See Table 1). We chose the above subsets because out of the list of ENC subsets, those seemed the closest to the Estonian GEC data domain.

At the sampling step, we apply filtering to the ENC sentences.

- We set an upper and a lower bound for the sentence length in tokens.
- The sentence should not start with punctuation other than variations of quotation marks.
- If the sentence starts with a word, it is capitalized.
- The sentence's last character must be a punctuation mark.
- There should be an even number of quotation marks in the sentence.

In addition to that, we removed unnecessary whitespaces from some Fiction subset sentences.

[9] https://github.com/TartuNLP/estgec/tree/main/Tartu_L2_corpus
[10] https://github.com/tlu-dt-nlp/EstGEC-L2-Corpus

3.2.2 Models

We generate synthetic errors using three models: GPT-3.5-Turbo, GPT-4-Turbo, and GPT-4. The specific model versions are `gpt-3.5-turbo-0613`, `gpt-4-1106-preview`, and `gpt-4-0613`, respectively. We utilize Microsoft Azure's OpenAI API to prompt GPT-3.5-Turbo and GPT-4. However, as of now, we have not obtained access to GPT-4-Turbo via Microsoft Azure's API, so we prompt GPT-4-Turbo using OpenAI's API instead.

We set the model temperature at 1 and allow up to three attempts to prompt the model in case of a failure. Additionally, we implement a 0.5-second cooldown period between prompts.

3.2.3 Prompting

Our goal is to introduce grammatical errors to a correct input sentence. To this extent, we create a prompt shown in Table 2. The English translation of the prompt can be seen in the appendix table 14. We prompt the model to generate such errors that Estonian language learners make. We specify the model to generate a variety of error types: spelling, grammar, word choice, word order, punctuation and style errors.

We apply two prompting techniques: zero- and few-shot prompting. For few-shot prompting technique we opt for four examples consisting of a source and a target sentence. Alongside the user prompt, we provide the model the a system prompt: "Muudad sisendtesksti vastavalt juhistele ning väljundtekstina tagastad muudetud teksti." (zero-shot) and "Muudad sisendtesksti vastavalt juhistele ning väljundtekstina tagastad muudetud teksti. Näed näiteid, mis demonstreerivad ülesannet." (few-shot). The system prompt translates to "You modify the input text according to the instructions and return the modified text as the output. You see examples that demonstrate the task." GPT models have learned to generate grammatically correct output, thus we provide as many examples as possible while not exceeding the token limit and keeping the resource consumption reasonable. We sample the examples from the UT L2 set using three methods:

1. random (`rand`),
2. Levenshtein distance (`lev`),
3. Sonar embedding cosine distance (`sonar`).

For `lev`, we calculate the Levenshtein distances between the input sentence and every source sentence in the UT L2. For `sonar`, we embed the input sentence and the UT L2 sentences with Sonar and calculate the cosine distance between the input sentence embedding and every source sentence embedding in UT L2. For both `lev` and `sonar`, we pick the top 4 closest source sentences to the input sentence. We pair the source sentence with its erroneous counterpart and use it as an example in the prompt.

See tables ref, ref and ref in the Appendix for an example of `rand`, `lev` and `sonar` for the same input sentence. It can be seen that the example sentences sampled with `lev` are orthographically similar while the example sentences sampled with Sonar are semantically similar.

Muuda sisendteksti, genereerides sinna vigu, mida võib teha eesti keele õppija. Väljundtekstina tagasta sisendtekst, kuhu oled genereerinud vead. Sisendteksti genereeri õigekirja-, grammatika-, sõnavaliku-, sõnajärje-, kirjavahemärgi- ning stiilivigu. Kui sisendtekstis on vigu, siis ära neid paranda, vaid genereeri vigu juurde. Sisendtekst: `{input}` Väljundtekst:

Table 2. Zero-shot prompt template.

Muuda sisendteksti, genereerides sinna vigu, mida võib teha eesti keele õppija. Väljundtekstina tagasta sisendtekst, kuhu oled genereerinud vead. Sisendteksti genereeri õigekirja-, grammatika-, sõnavaliku-, sõnajärje-, kirjavahemärgi- ning stiilivigu. Kui sisendtekstis on vigu, siis ära neid paranda, vaid genereeri vigu juurde. Ülesande kohta on neli näidet: Sisendtekst: `{correct}` Väljundtekst: `{incorrect}` Sisendtekst: `{correct}` Väljundtekst: `{incorrect}` Sisendtekst: `{correct}` Väljundtekst: `{incorrect}` Sisendtekst: `{correct}` Väljundtekst: `{incorrect}` Sisendtekst: `{input}` Väljundtekst:

Table 3. 4-shot prompt template.

3.2.4 Post-processing

We notice three main issues involving generating synthetic data with GPT that needs post-processing:

1. The output is not generated or is generated partly. Latter can happen due to API request failure, which a network or a rate limit error can cause.

2. The output is not generated due to safety model activation. Either caused by an example or input sentence containing unsafe words or a false activation.

3. There is noise in the output. Noise falls into three main categories: a) the model copies parts of the prompt to output; b) the model refuses to generate a response or generates a response that is unrelated to the task; c) the model generates a response related to the task but truncates large parts of the input sentence or hallucinates by adding more text to the input sentence.

We develop a post-processing pipeline with Python. In case of an API request failure or the safety model activation, we set the target sentence as the source sentence. To reduce noisy output, we compare the source sentence to the target sentence by sentence length in tokens and Levenshtein distance between the source and the target sentence. If GPT copies parts of prompts to the output, we extract the output sentence with Python's regular expression for string matching [11]. In this case, we ensure we do not copy an example sentence as the output. See GitHub[12] for more details.

3.2.5 Post-processing Results

I'll provide an overview how the data generation went. How many API errors we encountered? What are the post-processing stats?

3.3 Evaluation

This subsection gives an overview on the approaches used for the evaluation of synthetic errors.

3.3.1 Fine-tuning and Evaluating Models

One of our evaluation strategies involves fine-tuning the NLLB-200-Distilled-600M[13] model for grammatical error correction and assessing its performance on the EstGEC-L2 development set to evaluate how useful the data is for training the GEC systems. To achieve this, we fine-tune the NLLB-200-Distilled-600M on each generated dataset, including the human-sourced UT-L2 set. For training we use the Fairseq library (Ott et al., 2019). We train the model to translate from Estonian to Estonian, using the language codes Est0_Latn for the source and Est_Latn for the target in the NLLB language codes

[11] https://docs.python.org/3/library/re.html
[12] https://github.com/nitram-v/est-gec-synth-by-prompting
[13] https://github.com/facebookresearch/fairseq/tree/nllb

but use language code Est_Latn both as source and target token. To assess the generated hypotheses, we perform model evaluations separately from Fairseq's automatic evaluation using an M^2 scorer script modified for Estonian. We evaluate every epoch checkpoint and report development results on the epoch with the highest $F_{0.5}$-score.

We apply the standard NLLB preprocessing, which includes normalization and SentencePiece (Kudo and Richardson, 2018) tokenization. We utilize a SentencePiece model with a size of 256,000, trained by (Team et al., 2022). For normalization, we use a standard script[14].

We train the NLLB model on each dataset for 25 epochs with a learning rate of 0.0001, employing the Adam optimizer (Kingma and Ba, 2015) and an inverse square root learning rate scheduler. Our models are trained and evaluated at the University of Tartu's High Performance Computing Center (HPC)(University of Tartu, 2018) on Nvidia's Tesla a100 GPU with 40 GB of vRAM, with training time averaging around one hour.

3.3.2 Human Evaluation

We manually tag the synthetic error sentences (human evaluation) in a modified M^2 format. We look at both the correct (source) and the (synthetic) error (target) sentence, which adds the possibility of detecting hallucinating, omitting, and similar behavior. However, if we only look at the erroneous sentence, we do not know if the meaning of the original sentence has changed or if GPT has fixed the mistakes present in the source sentence. In a standard setting, the evaluator only sees the erroneous sentence.

During evaluation, we notice cases where the GPT model generates new content for the sentence or removes words crucial for understanding the meaning of the sentence. A synonym replaces some words. Another optional infliction replaces some inflictions. GPT might also correct an erroneous input sentence. Some GPT-generated spelling errors made the original word unrecognizable.

We introduce new tags for the aforementioned cases:

- HALL - hallucinating;
- SYN - synonym;
- O - optional, but correct;
- INACC - there is a mistake in the correct sentence that GPT fixes;
- UNREC - such spelling error that the original word is not recognizable.

[14] https://github.com/pluiez/NLLB-inference/blob/main/preprocess/normalize-punctuation.perl

Note that HALL stands for both omitting and hallucinating depending on the operation: HALL tag with U indicates hallucinating and with M omitting text. M^2 format modified for Estonian with the annotation guide is also used to label the EstGEC-L2 test sets.

Model	P	R	$F_{0.5}$
GPT-3.5-Turbo	**56.47**	36.36	50.85
GPT-4-Turbo	54.03	**42.20**	51.16
GPT-4	55.80	41.42	**52.18**

Table 4. $F_{0.5}$-scores for NLLB-600M models fine-tuned on synthetic errors generated by GPT-3.5-Turbo, GPT-4-Turbo and GPT-4.

4 Results

In this chapter, we present the results of the fine-tuning and human evaluation. We generated errors with three different GPT versions: GPT-3.5-Turbo, GPT-4-Turbo and GPT-4. We generated errors in two sets of source sentences: UT-L2 train (ut) and ENC-GEC (enc). We applied four different prompting methods: zero-shot (zs), 4-shot prompting with random (rand), Levenshtein (lev) distance and sonar (sonar) examples. In total, we generated 15 sets of synthetic data.

4.1 Fine-tuning

4.1.1 Error Generation Models

Table 4 displays the development set results for NLLB-600M models that were fine-tuned on three sets of synthetic data generated by GPT-3.5-Turbo, GPT-4-Turbo, and GPT-4. The prompting method used was four-shot prompting with randomly sampled examples. Errors were introduced into the correct sentences of the UT L2 corpus. The primary purpose of this section is to compare the efficacy of different GPT models in generating errors.

The results indicate a significant difference in recall between GPT-3.5-Turbo and both GPT-4 and GPT-4-Turbo, with GPT-4 models showing significantly higher recall and GPT-3.5-Turbo showing a higher precision. A higher recall suggests that the model corrected more grammatical errors overall. Conversely, higher precision implies more accurate error corrections.

Since the data from GPT-3.5-Turbo achieved the highest precision, its $F_{0.5}$-score is on par with those of GPT-4 and GPT-4-Turbo. Considering both cost and generation time, GPT-3.5-Turbo offers the best value for money. Table 5 provides a detailed comparison of the time and financial costs associated with each model. Notably, GPT-3.5-Turbo is significantly more cost-effective than GPT-4-Turbo and GPT-4, being 7 times less expensive than GPT-4-Turbo and 20 times less than GPT-4.

Due to considerations of price, time consumption, and performance, the subsequent experiments—specifically the Seed Corpus and Prompting sections were primarily con-

Model	Input Tokens	Output Tokens	Cost in EUR	Time
GPT-3.5-Turbo	5.45M	0.34M	8.15	2 h 51 min
GPT-4-Turbo	5.42M	0.33M	58.93	8 h 47 min
GPT-4	5.46M	0.33M	168.84	13 h 27 min

Table 5. The number of input and output tokens alongside money and time consumption for generating errors to UT L2 corpus of 8,921 sentences. Models are GPT-3.5-Turbo, GPT-4-Turbo and GPT-4 in a 4-shot setting with random examples. The cost is calculated according to OpanAI's pricing `https://openai.com/api/pricing/`

ducted using GPT-3.5-Turbo. The efficacy and synthetic error quality of GPT-4 and GPT-4-Turbo is discussed in the Discussion subsection. Although these models may generate a more diverse range of synthetic grammatical errors, they significantly increase both cost and time consumption.

4.1.2 Seed Corpus

This subsection examines the impact of generating errors in data from different domains. The training and development sets consist of sentences collected from students' essays, which share the same domain. In addtion, another subset from the ENC is considered due to the challenges in collecting sentences from student essays. This includes equal Fiction, Wiki, and Web subsets from the ENC.

Table 6 presents the results of NLLB-600M models trained on errors introduced into different data domains using four-shot prompting with random examples. Table 6 compares different models across these domains. The $F_{0.5}$-score significantly decreased when generating errors to source sentences from a different domain. There was a decline in both precision and recall. Notably, with GPT-3.5-Turbo, there was a larger drop in recall than with GPT-4-Turbo. With GPT-4-Turbo, the decreases in precision and recall were equal. A decrease in recall can indicate that there are fewer errors per sentence. There could also be a mismatch in the vocabulary used in the sentences.

To explore the possible domain mismatch, we calculated the similarity of the input sentences to example sentences for ENC and UT-L2 input sentences. We calculated Levenshtein distances between the input sentence and each source sentence of the examples (see Figure 3). We repeated the same for Sonar embeddings but with cosine distances (see Figure 4). Not surprisingly, there is a noticeable difference between the similarity histograms when comparing ENC to UT-L2. We noticed that there can be multiple versions of the same erroneous sentence in the UT-L2 corpus. Interestingly, there was a more noticeable difference between ENC and UT-L2 when comparing cosine distances between Sonar embeddings, which indicates semantical similarity.

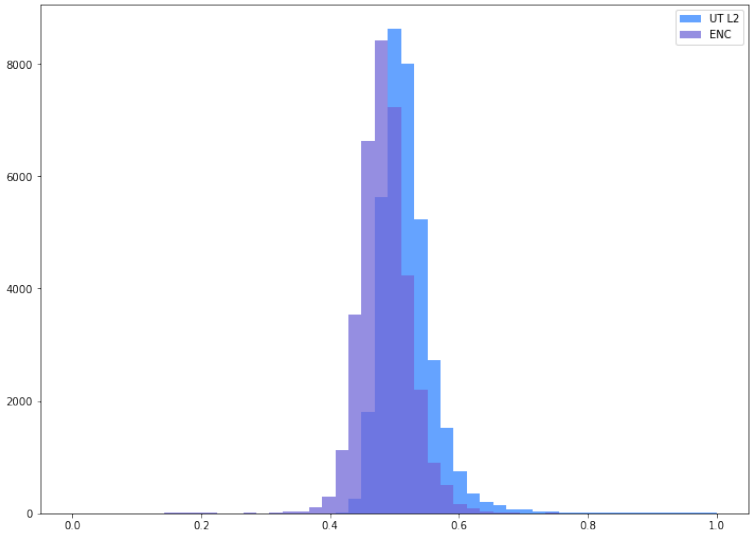

Figure 3. Histograms of Levenshtein distances between the ENC and UT-L2 input sentences and the source sentences of the examples.

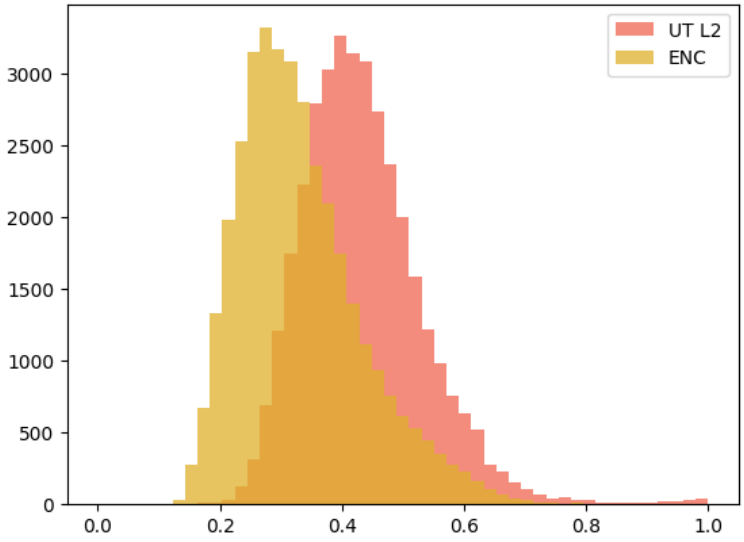

Figure 4. Histograms of the cosine distances between the ENC and UT-L2 input sentence Sonar embeddings and the source sentence Sonar embeddings of the examples.

Dataset/Model	GPT-3.5-Turbo			GPT-4-Turbo		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$
ut-rand	56.47	36.36	50.85	54.03	42.20	51.16
enc-rand	52.69	29.11	45.35	48.41	36.60	45.47

Table 6. $F_{0.5}$-scores for NLLB-600M models fine-tuned on synthetic errors generated by GPT-3.5-Turbo and GPT-4-Turbo with one additional data domain, a subset of Wiki, Web and Fiction from ENC. Prompting method was four-shot prompting with random examples.

4.1.3 Prompting Method

In this subsection we compare different prompting methods: zero-shot and few-shot prompting with four examples yielded by three different sampling strategies: 1) random, 2) closest Levenshtein distance and 3) closest cosine distance of Sonar vectors. Table 7 presents the results of using different prompting methods with both GPT-3.5-Turbo and GPT-4-Turbo.

Surprisingly, the zero-shot prompting method yielded comparable $F_{0.5}$-scores to 4-shot prompting. GPT-4-Turbo was less affected by the absence of examples, achieving almost on par $F_{0.5}$-score to 4-shot with random examples. There was a more significant, roughly 3-point difference between zero-shot and 4-shot with GPT-3.5-Turbo. When considering precision and recall, there was a 3 and 2-point drop, respectively. On the other hand, GPT-4-Turbo achieved the highest recall across the experiments with zero-shot prompting. When comparing zero-shot to 4-shot with random examples, there is a 2-point increase in recall but a 2-point decrease in precision. Interestingly, each GPT-4-Turbo prompting strategy achieved 1-3 points higher recall than human data. We further explore explanations for the high recall in Section 4.2.

When shifting the attention to ENC, we notice similar trends with zero-shot prompting with GPT-3.5-Turbo, but there is a different case for GPT-4-Turbo. Zero-shot prompting yielded the highest $F_{0.5}$-score across the ENC $F_{0.5}$-scores. There was a less significant drop in precision with a noticeably higher recall when compared to 4-shot methods.

Except for ENC GPT-4-Turbo, Levenshtein achieved the highest $F_{0.5}$-score out of the three 4-shot prompting methods. Levnshtein preserved a high precision with a slight increase in recall. Examples with the closest Sonar embedding examples do not improve Levenshtein scores.

Dataset/Model	GPT-3.5-Turbo			GPT-4-Turbo		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$
ut-zs	53.43	34.00	47.95	52.86	44.78	51.02
ut-rand	56.47	36.36	50.85	54.03	42.20	51.16
ut-lev	58.59	37.25	52.57	55.25	43.09	52.30
ut-sonar	58.95	35.81	52.20	-	-	-
enc-zs	51.17	28.34	44.07	48.36	38.11	45.89
enc-rand	52.69	29.11	45.35	48.41	36.60	45.47
enc-lev	51.85	30.43	45.45	48.77	35.17	45.27
enc-sonar	52.15	29.71	45.31	-	-	-

Table 7. $F_{0.5}$-scores for NLLB-600M models fine-tuned on synthetic errors generated by GPT-3.5-Turbo and GPT-4-Turbo with different prompting methods on UT L2 domain data.

Dataset	P	R	$F_{0.5}$
UT L2	58.49	41.33	54.01
ut-lev-3.5-turbo	58.59	37.25	52.57
ut-lev-4-turbo	55.25	43.09	52.30
enc-lev-3.5-turbo	51.85	30.43	45.45
enc-zs-4-turbo	48.36	38.11	45.89

Table 8. $F_{0.5}$-scores of models fine-tuned on UT L2 human corpus and synthetic errors generated by various methods chosen by the highest $F_{0.5}$-score.

4.1.4 Synth. vs. Human Data

In this subsection, we compare the highest-performing error generation methods based on $F_{0.5}$-scores from both data domains to human data. We present the results in 8. Both GPT-3.5-Turbo and GPT-4-Turbo are represented. The highest $F_{0.5}$-score was achieved with GPT-3.5-Turbo on UT L2 in a 4-shot setting with Levenshtein examples. There is a 1.5-point difference between human data and the highest-performing GPT synthetic error method.

4.2 Human Evalutation

In this subsection, we present the results of the human evaluation. For this evaluation, we selected five error generation methods: ut-rand-gpt-3.5-turbo, ut-lev-gpt-3.5-turbo, enc-rand-gpt-3.5-turbo, ut-zs-gpt-4-turbo, and ut-rand-gpt-4-turbo. We randomly sampled 50 indices and annotated the sentences at these indices using M^2 tags, along with GPT-specific tags for identifying problematic errors. We chose these sets to encompass all comparison strategies while minimizing the number of sets, as human evaluation is time-consuming. We compared different model versions, prompting methods (ranging from zero-shot to few-shot and various few-shot example sampling methods), data domains, and synthetic errors against human errors. The sets can be accessed from GitHub [15].

Data	Errors per sent	Errors in total
ut-human	3.34	167
ut-zs-4-turbo	7.88	394
ut-rand-4-turbo	5.00	250
ut-rand-3.5-turbo	2.78	139
enc-rand-3.5-turbo	2.70	135
ut-lev-3.5-turbo	2.28	114

Table 9. The average number of errors per sentence alongside the total number of errors.

Table 9 presents the average number of errors in a sentence alongside the total number of errors per set. GPT-4-Turbo generated more errors per sentence than GPT-3.5-Turbo, with GPT-4-Turbo yielding the highest number of errors per sentence with zero-shot. The average number of human errors per sentence falls between GPT-3.5-Turbo and GPT-4-Turbo. Interestingly, the Levenshtein example sampling method yielded the smallest number of errors per sentence. Different data domain to GEC (ENC) resulted in a similar average to the GEC domain.

Table 10 displays the percentages of problematic errors (edits) for each set. We observed that the percentages varied significantly across different types of errors, making it challenging to draw definitive conclusions. However, the overall percentage of problematic errors was higher with GPT-3.5-Turbo than with GPT-4-Turbo. Additionally, zero-shot prompting with GPT-4-Turbo resulted in the lowest total percentage of problematic errors. Yet, the total number of problematic errors was comparable to other sets (as shown in Figure 6), largely due to a high number of spelling and nominal form errors.

Table 11 shows the percentages of each set's main error types, revealing variation across sets. In nearly all sets except for the zero-shot with GPT-4-Turbo, we observed that the most prevalent error type was R:NOM:FORM. Additionally, the GPT-3.5-Turbo

[15] https://github.com/nitram-v/est-gec-synth-by-prompting

Data	O	HALL	SYN	INACC	UNREC	Total %
ut-human	0.0	0.0	0.0	**1.8**	0.0	1.8
ut-lev-3.5-turbo	10.53	**14.04**	5.26	1.75	4.39	35.97
ut-rand-3.5-turbo	8.63	6.47	**13.67**	3.6	0.72	33.09
enc-rand-3.5-turbo	5.93	**9.63**	3.7	**9.63**	3.7	32.59
ut-rand-4-turbo	**11.2**	6.0	6.0	1.6	0.8	25.6
ut-zs-4-turbo	**4.57**	2.03	4.57	0.76	0.76	12.69

Table 10. Percentages of problematic edits with the highest error percentage of the set in bold. Note that "HALL" encompasses both hallucinating and truncating of text that changes the text's original meaning.

prompted the UT L2 set with random examples that showed as many R:LEX errors as R:NOM:FORM errors. While R:LEX was the second most common error type in UT L2 human sentences, GPT-generated sentences tended to produce more spelling errors, with GPT-4-Turbo zero-shot prompted UT L2 having R:SPELL as the most frequent error type.

Next, we compared the error distributions of different error generation methods side-by side. For comparison, we also included UT L2 human sentences on each figure.

Figure 5 compares the distribution of errors generated to UT L2 by GPT-3.5-Turbo and GPT-4-Turbo in 4-shot setting with random examples. We noticed that GPT-4-Turbo generated sentences had more errors in them, especially R:NOM:FORM and R:VERB:FORM erros. Meanwhile GPT-3.5-Turbo generated sentences had generally less errors than UT L2 human sentences. With GPT-4-Turbo generating more errors, the number of problematic edits is slightly higher with GPT-4-Turbo than GPT-3.5-Turbo.

Figure 6 compares the distribution of errors generated to UT L2 by GPT-4-Turbo in zero-shot and 4-shot settings with random examples. Zero-shot yielded significantly more spelling errors than 4-shot and around ten times more than in the UT L2 human set. Compared to the 4-shot, there were more whitespace, verb and nominal form errors. With zero-shot, the number of problematic errors increased a little.

Figure 7 compares the distribution of errors generated to UT L2 by GPT-3.5-Turbo in a 4-shot setting with random and Levenshtein examples. We noticed that except for the spelling errors with random examples, there were fewer errors for each type in sets generated by GPT.

Figure 8 compares the distribution of errors generated to UT L2 and ENC by GPT-3.5-Turbo in a 4-shot setting with random examples. Regardless of domain, the error distribution between errors generated to ENC and UT L2 are similar. For ENC, there were slightly more punctuation, whitespace and case errors. These can be the errors that GPT fixed in the ENC source sentence, making these errors problematic, although the

Error	ut-human	enc-rand-3.5-turbo	ut-lev-3.5-turbo	ut-rand-3.5-turbo	ut-rand-4-turbo	ut-zs-4-turbo
R:NOM:FORM	**32.34**	**23.7**	**26.32**	**25.18**	**44.0**	31.47
R:LEX	16.17	16.3	18.42	**25.18**	13.2	11.68
R:WO	13.77	7.41	10.53	9.35	4.4	2.03
R:VERB:FORM	7.78	7.41	10.53	7.19	16.4	12.18
R:SPELL	7.19	17.78	13.16	17.27	10.8	**34.01**
M:PUNCT	5.99	2.96	1.75	2.16	2.0	1.27
M:LEX	5.39	5.19	4.39	5.04	0.8	0.51
U:LEX	5.39	6.67	10.53	2.88	3.6	1.52
R:WS	4.19	2.22	3.51	2.88	4.4	4.57
U:PUNCT	1.2	4.44	0.88	1.44	0.4	0.51
R:PUNCT	0.6	3.7	0.0	0.72	0.0	0.25
R:CASE	0.0	2.22	0.0	0.72	0.0	0.0

Table 11. Percentages of the main error types out of total errors per set with the highest error percentage of the set in bold. Errors with problematic edits were included when calculating the percentages.

total number of problematic errors is similar.

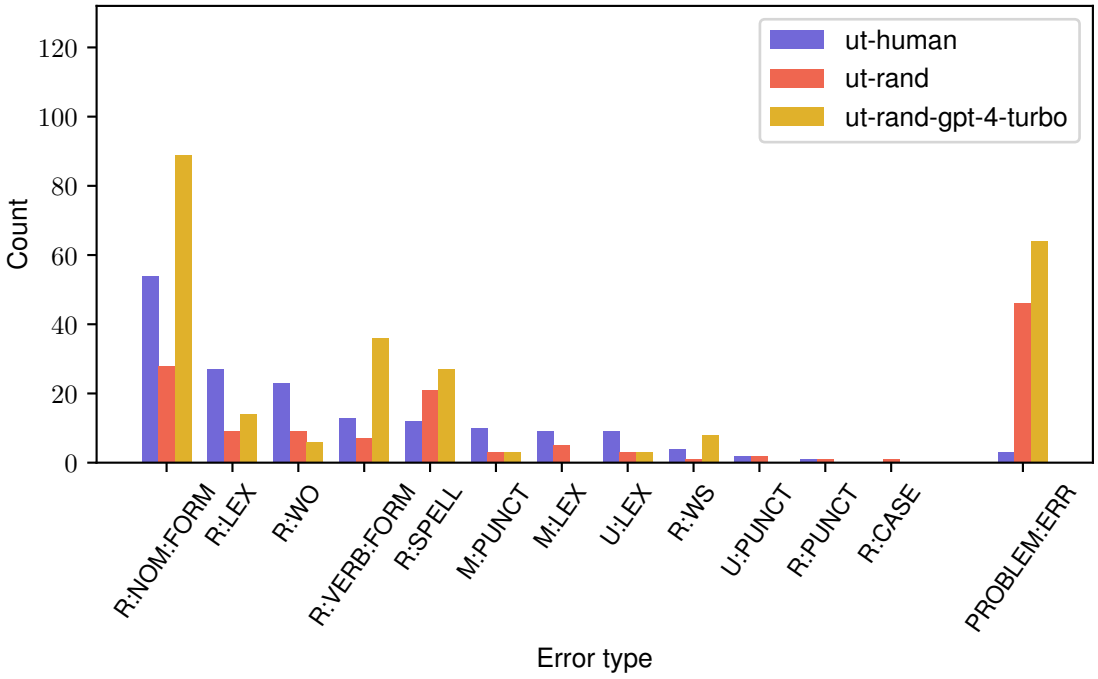

Figure 5. The distribution of errors generated to UT L2 by GPT-3.5-Turbo and GPT-4-Turbo in a 4-shot setting with random examples. If any main error types contained a problematic edit, we regarded it as PROBLEM:ERR. "ut-rand" on the legend should be read as "ut-rand-3.5-turbo".

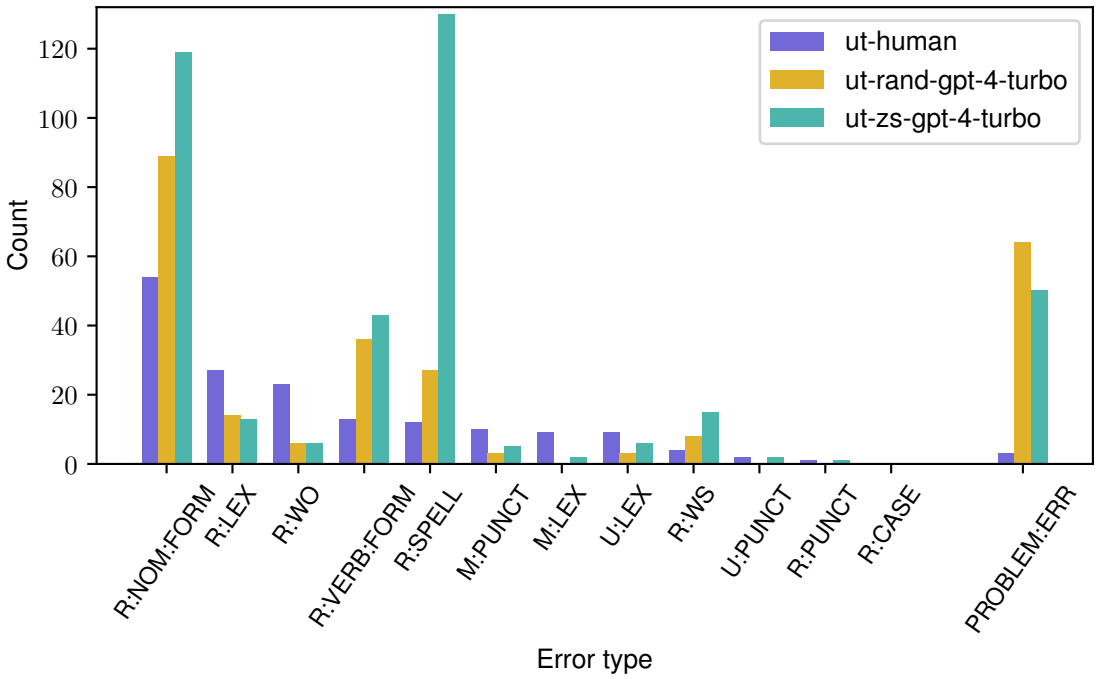

Figure 6. The distribution of errors generated to UT L2 by GPT-4-Turbo in zero-shot and 4-shot settings with random examples. If any main error types contained a problematic edit, we regarded it as PROBLEM:ERR.

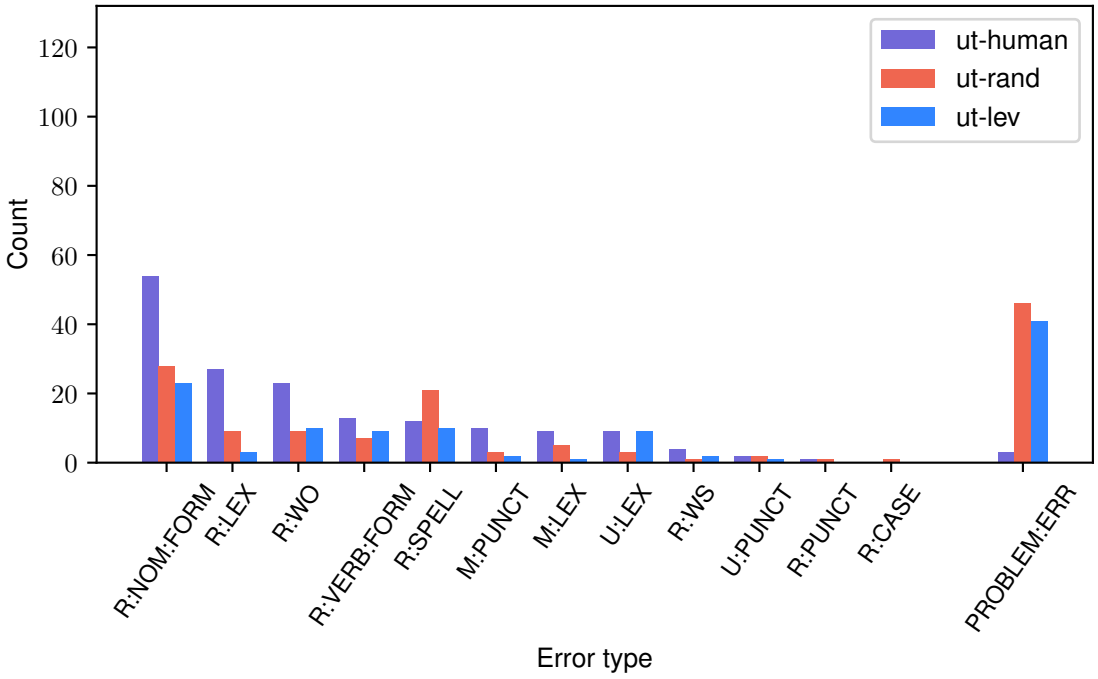

Figure 7. The distribution of errors generated to UT L2 by GPT-3.5-Turbo in a 4-shot setting with random and Levenshtein examples. If any main error types contained a problematic edit, we regarded it as PROBLEM:ERR. On the legend, "ut-rand" should be read as "ut-rand-3.5-turbo" and "ut-lev" as "ut-lev-3.5-turbo".

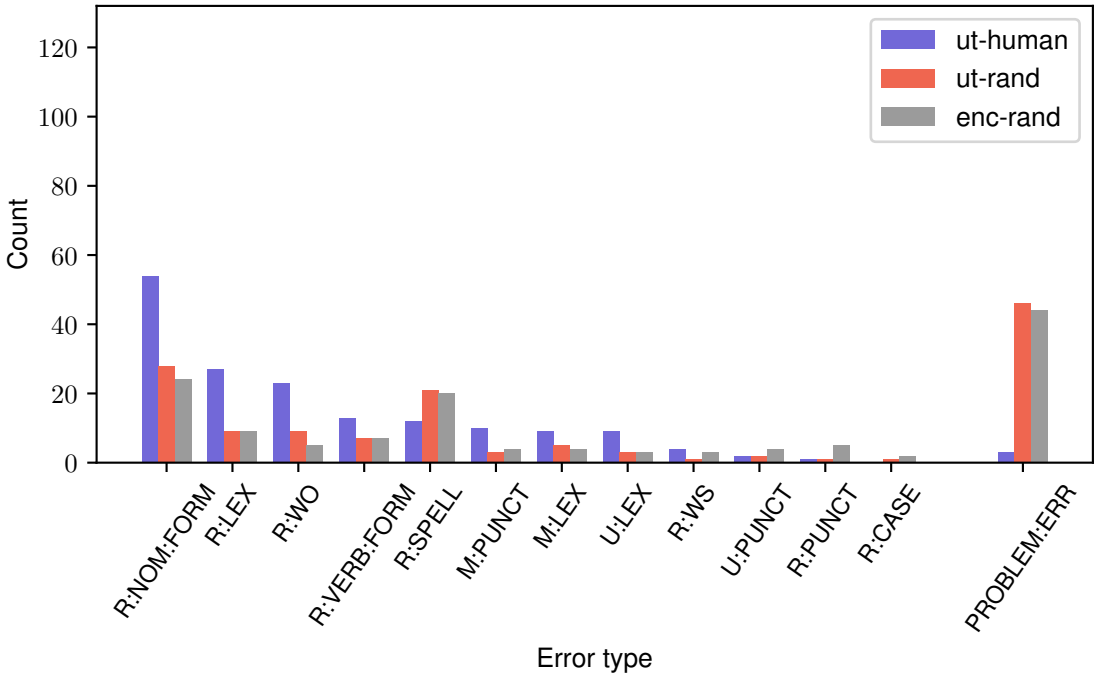

Figure 8. The distribution of errors generated to UT L2 and ENC by GPT-3.5-Turbo in a 4-shot setting with random examples. If any main error types contained a problematic edit, we regarded it as PROBLEM:ERR. On the legend, "ut-rand" should be read as "ut-rand-3.5-turbo" and "enc-rand" as "enc-rand-3.5-turbo".

5 Discussion

The results indicated that the model, prompt, and domain data choice significantly influenced error generation. GPT-4-Turbo produced a higher number of grammatical errors per sentence. Notably, NLLB-200-Distilled-600M, when fine-tuned on sets with a high incidence of grammatical errors generated by GPT-4-Turbo (especially in a zero-shot setting), achieved high recall on the development set. Conversely, models trained on datasets with fewer errors generated by GPT-3.5-Turbo in a 4-shot setting achieved higher precision and, consequently, higher $F_{0.5}$-scores. High recall can lead to low precision as the model may make more edits to an input sentence and vice versa. Depending on the specific requirements of the downstream GEC system, either scenario could be beneficial.

$F_{0.5}$-scores on generated data were lower than on human data. Interestingly, zero-shot with GPT-4-Turbo yielded a noticeably higher recall than human data but showed a significant decrease in precision. The results deteriorated on an out-of-domain synthetic GEC dataset, although the error type distribution remained similar across domains. We speculate that domain mismatches contribute to lower scores. Additionally, out-of-domain sentences may contain grammatical errors, and alongside introducing new grammatical errors to an erroneous sentence, GPT can correct the present errors.

We further discuss these observations in the Limitations and Future Work subsections.

5.1 Limitations

5.1.1 Domain Mismatch

Sentences from different domains contain different vocabulary and sentence structures. We collected the sentences from ENC Fiction, Web and Wikipedia splits. UT L2 sentences are from essays. We plotted the similarity distributions between input sentences and examples to show the domain mismatch. Some sentences within the UT L2 train set are very similar; possibly, there are multiple versions of the same sentence with slight differences within the set. This could skew the results of the generated UT L2 sets when sampling sentences based on Levenshtein distance or cosine distance on Sonar embeddings. Out-of-domain sentences might not see sentences as similar to input, which could disadvantage ENC. In addition, UT L2 train and dev are of the same domain.

5.1.2 Safety Model

OpenAI API runs the prompts through a safety model. The API returns no generated response if the safety model flags the prompt as potentially unsafe. During error generation, a subset of sentences did not receive a response due to the prompt getting flagged. Even though the prompt's instructions were not unsafe, the example sentences or input

sentences can contain unsafe content. We also noticed false flagging. Unresponded sentences raise the problem that sentences with certain words will be missing from the synthetic dataset, which might impact the model's performance.

Moreover, Azure OpenAI API seems to have a stricter safety model because a larger number of sentences were flagged compared to OpenAI API.

5.1.3 Problematic Edits

As we showed, some of the errors GPT generated were problematic. When training a GEC model on a dataset that contains such edits, the model could learn unwanted behaviour, which in turn affects the scores on development set.

5.1.4 Benchmarking

The M^2 benchmark is rigid: it only allows a fixed number of feasible corrections for a given sentence with errors. In reality, there can be many feasible corrections for an erroneous sentence. One type of problematic error GPT generated was an optional edit, such as an optional word form, replacing a word with a synonym, changing the word order to another viable order, etc. Again, the GEC model could learn this behavior and suggest optional edits but receive a low score on the benchmark.

5.2 Future Work

Future work holds engineering a prompt that yields less problematic edits. Evaluation can be done similarly to how we performed human evaluation. Also, different temperature values should be tested as lower values can result in more reserved edits, which could reduce hallucinations. One more possible solution for hallucinations could be to filter such sentence-pairs out by prompting an LLM whether the corrections are valid, when there is no outside context provided about the sentence. Another possibility is to prompt another LLM for error generation or further explore fine-tuning an LLM for artificial error generation as done by Luhtaru et al. (2024b).

Future work should also focus on extracting or generating sentences that align more closely with the GEC domain. This could be achieved by selecting sentences from the ENC using sonar vectors, with selection based on a cosine similarity threshold. However, this approach raises questions about its effectiveness. Since the UT L2 corpus is not exhaustive, we might see improved results on the test/development sets, but the performance gain might not extend beyond these confines. The applicability of this method also depends on the specific downstream task. For instance, should the correction focus solely on sentences written in an essay style, or should it extend to academic texts, Wikipedia pages, and other forms? Additionally, obtaining human-generated data from these various domains would be beneficial.

To conclude, this work can be extended to generating a large synthetic dataset by improving the error generation method, whether by prompt-engineering, fine-tuning or finding sentences closer to the domain. To do that, the specific downstream GEC task should be defined, and its use cases analyzed in detail to avoid domain mismatch.

6 Conclusion

In this master's thesis, we experimented with generating human-like Estonian GEC data using GPT prompts. We produced 15 sets of synthetic data utilizing different GPT versions, prompting methods, 4-shot example sampling methods, and source sentences both within and outside the GEC domain. To assess synthetic errors against human errors, we introduced errors into the correct sentences of the UT L2 train set (within the GEC domain). We evaluated the synthetic errors by 1) fine-tuning an MT model (NLLB-200-Distilled-600M) on each synthetic set, and 2) conducting human evaluations by annotating 50 sentences from five synthetic sets. Each fine-tuned MT model was evaluated on the L2 development set, comparing precision, recall, and $F_{0.5}$-score across the models. For human evaluations, we annotated the sets using the main error types from M^2 and introduced five additional error types for GPT-specific problematic edits. We repeated the fine-tuning and human evaluations on human data.

Our results indicated that within the GEC domain, MT scores based on synthetic data were comparable to those achieved with human data, with the best $F_{0.5}$-score being only 1.5 points lower than that on human data. Additionally, we observed that the MT model trained on a set generated by GPT-4-Turbo in a zero-shot setting achieved a higher recall compared to human data. However, data from outside the GEC domain yielded significantly worse results, likely due to domain mismatch. Human evaluations revealed a high number of problematic edits in the GPT-generated sets. We also noticed that different versions of models generate varying number of errors.

Answering the questions we raised in the Introduction, we note that this work opens many possibilities for future work:

1. Prompting GPT is indeed a viable technique for generating human-like synthetic grammatical errors, but future research should be done in order to reduce problematic edits.

2. Different models generated different numbers of errors per sentence, with GPT-4-Turbo generating more errors than GPT-3.5-Turbo.

3. Few-shot yielded fewer errors per sentence than zero-shot. Examples sampled with Levenshtein distance showed the highest $F_{0.5}$-scores.

4. Out-of-domain sentences had a significant drop in $F_{0.5}$-scores while the error distribution was relatively similar to the synthetic GEC domain. We showed that the drop in scores can be caused by domain mismatch. Finding sentences similar to the GEC domain is future research.

Two primary areas for future research include reducing the number of problematic errors in the generated sentences and more effectively aligning source sentences with the

GEC domain. Additionally, it is crucial for future efforts to precisely define and analyze the downstream GEC task to prevent domain mismatches.

References

C. Brockett, W. B. Dolan, and M. Gamon. Correcting ESL errors using phrasal SMT techniques. In N. Calzolari, C. Cardie, and P. Isabelle, editors, *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 249–256, Sydney, Australia, July 2006. Association for Computational Linguistics. doi: 10.3115/1220175.1220207. URL https://aclanthology.org/P06-1032.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

C. Bryant, M. Felice, and T. Briscoe. Automatic annotation and evaluation of error types for grammatical error correction. In R. Barzilay and M.-Y. Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1074. URL https://aclanthology.org/P17-1074.

C. Bryant, Z. Yuan, M. R. Qorib, H. Cao, H. T. Ng, and T. Briscoe. Grammatical error correction: A survey of the state of the art. *Computational Linguistics*, pages 643–701, Sept. 2023. doi: 10.1162/coli_a_00478. URL https://aclanthology.org/2023.cl-3.4.

S. Coyne, K. Sakaguchi, D. Galvan-Sosa, M. Zock, and K. Inui. Analyzing the performance of gpt-3.5 and gpt-4 in grammatical error correction, 2023.

D. Dahlmeier and H. T. Ng. Better evaluation for grammatical error correction. In E. Fosler-Lussier, E. Riloff, and S. Bangalore, editors, *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada, June 2012. Association for Computational Linguistics. URL https://aclanthology.org/N12-1067.

R. Dale and A. Kilgarriff. Helping our own: The hoo 2011 pilot shared task. In

Proceedings of the 13th European Workshop on Natural Language Generation, pages 242–249, 2011.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

P.-A. Duquenne, H. Schwenk, and B. Sagot. SONAR: sentence-level multimodal and language-agnostic representations, 2023. URL https://arxiv.org/abs/2308.11466.

M. Erelt. Keeletüpoloogiast. *Oma Keel*, 1:5–13, 2005.

M. Erelt, T. Erelt, and K. Ross. *Eesti keele käsiraamat*. Tallinn: Eesti Keele Instituut, 2020.

T. Erelt, T. Leemets, S. Mäearu, and M. Raadik. *Eesti õigekeelsussõnaraamat ÕS 2018*. Tallinn: Eesti Keele Sihtasutus, 2018.

R. Grundkiewicz, M. Junczys-Dowmunt, and K. Heafield. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In H. Yannakoudakis, E. Kochmar, C. Leacock, N. Madnani, I. Pilán, and T. Zesch, editors, *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263, Florence, Italy, Aug. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4427. URL https://aclanthology.org/W19-4427.

M. Junczys-Dowmunt, R. Grundkiewicz, S. Guha, and K. Heafield. Approaching neural grammatical error correction as a low-resource machine translation task. In M. Walker, H. Ji, and A. Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 595–606, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1055. URL https://aclanthology.org/N18-1055.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015. URL https://arxiv.org/abs/1412.6980.

K. Koppel and J. Kallas. Eesti keele ühendkorpuste sari 2013–2021: mahukaim eestikeelsete digitekstide kogu. *Eesti Rakenduslingvistika Ühingu aastaraamat Estonian Papers in Applied Linguistics*, 18:207–228, 06 2022. doi: 10.5128/ERYa18.12.

T. Kudo and J. Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In E. Blanco and W. Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language*

Processing: System Demonstrations, pages 66–71, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL https://aclanthology.org/D18-2012.

V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

A. Luhtaru. Low-resource grammatical error correction via synthetic pre-training and monolingual zero-shot translation, 2022.

A. Luhtaru, E. Korotkova, and M. Fishel. No error left behind: Multilingual grammatical error correction with pre-trained translation models. In Y. Graham and M. Purver, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1209–1222, St. Julian's, Malta, Mar. 2024a. Association for Computational Linguistics. URL https://aclanthology.org/2024.eacl-long.73.

A. Luhtaru, T. Purason, M. Vainikko, M. Del, and M. Fishel. To err is human, but llamas can learn it too, 2024b.

A. Luhtaru, M. Vainikko, K. Liin, K. Allkivi-Metsoja, J. Kippar, P. Eslon, and M. Fishel. Autocorrect for estonian texts: final report from project ektb25, 2024c.

C. Napoles, K. Sakaguchi, and J. Tetreault. JFLEG: A fluency corpus and benchmark for grammatical error correction. In M. Lapata, P. Blunsom, and A. Koller, editors, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 229–234, Valencia, Spain, Apr. 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-2037.

H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant. The CoNLL-2014 shared task on grammatical error correction. In H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant, editors, *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-1701. URL https://aclanthology.org/W14-1701.

OpenAI. Gpt-4 technical report, 2023.

M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback, 2022.

A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

I. Rummo and K. Praakli. TÜ eesti keele (võõrkeelena) osakonna õppijakeele tekstikorpus [the language learner's corpus of the department of estonian language of the university of tartu]. In *EAAL 2017: 16th annual conference Language as an ecosystem, 20-21 April 2017, Tallinn, Estonia: abstracts, 2017, p. 12-13*, 2017.

N. Team, M. R. Costa-jussà, J. Cross, O. Çelebi, M. Elbayad, K. Heafield, K. Heffernan, E. Kalbassi, J. Lam, D. Licht, J. Maillard, A. Sun, S. Wang, G. Wenzek, A. Youngblood, B. Akula, L. Barrault, G. M. Gonzalez, P. Hansanti, J. Hoffman, S. Jarrett, K. R. Sadagopan, D. Rowe, S. Spruit, C. Tran, P. Andrews, N. F. Ayan, S. Bhosale, S. Edunov, A. Fan, C. Gao, V. Goswami, F. Guzmán, P. Koehn, A. Mourachko, C. Ropers, S. Saleem, H. Schwenk, and J. Wang. No language left behind: Scaling human-centered machine translation, 2022.

University of Tartu. Ut rocket, 2018.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

W. Zhao, L. Wang, K. Shen, R. Jia, and J. Liu. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 156–165, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1014. URL https://aclanthology.org/N19-1014.

Appendix

I. M^2 Tags

Error tag	Meaning	Example
R:SPELL	Spelling error	soobib -> sobib
R:CASE	Capitalization error	Juuli -> juuli
R:WS	Whitespace error	igalpool -> igal pool
R:NOM:FORM	Nominal form error	kallis -> kallid (Sing -> Plur)
R:VERB:FORM	Verb form error	tegeleb -> tegeles (Pres -> Past)
R:LEX	Word choice error	ilusasti -> ilus (ADV -> ADJ)
R:PUNCT	Punctuation choice error	Kohtumiseni. -> Kohtumiseni!
R:WO	Word order error	üldse polnud -> polnud üldse
M:LEX	Missing word(s)	See väga ilus linn -> See on väga ilus linn
U:LEX	Unnecessary word(s)	auto välimus on punane -> auto on punane
U:PUNCT	Unnecessary punctuation	laupäeval, kell 10 -> laupäeval kell 10

Table 12. Overview of the main error types, their meanings and examples. The table is from EstGEC-L2 GitHub (`https://github.com/tlu-dt-nlp/EstGEC-L2-Corpus`)

Table 13. Overview of complex error tags, their meanings and examples. The table is from EstGEC-L2 GitHub (`https://github.com/tlu-dt-nlp/EstGEC-L2-Corpus`)

Error tag	Meaning	Example
R:SPELL:CASE	Spelling and capitalization error	Vannalinnas -> vanalinnas
R:WS:SPELL	Whitespace and spelling error	liimik koht -> lemmikkoht
R:WS:CASE	Whitespace and capitalization error	Kontserdi majas -> kontserdimajas
R:WS:NOM:FORM	Whitespace and nominal form error	kogupäev -> kogu päeva (Nom -> Gen)

Continued on next page

Table 13 continued from previous page

Error tag	Meaning	Example
R:WS:NOM:FORM:SPELL	Whitespace, nominal form and spelling error	politika uudiseid -> poliitikauudised (Par -> Nom)
R:WS:NOM:FORM:CASE	Whitespace, nominal form and capitalization error	cv online -> CV-Online'i (Nom -> Gen)
R:NOM:FORM:SPELL	Nominal form and spelling error	ekskursioni ekskursiooni -> ekskursioonile (Gen/Par -> All)
R:NOM:FORM:CASE	Nominal form and capitalization error	tartu -> Tartut (Nom -> Par)
R:NOM:FORM:SPELL:CASE	Nominal form, spelling and capitalization error	Sobrad Sõbrad -> sõpradega (Nom -> Com)
R:VERB:FORM:SPELL	Verb form and spelling error	kaisin käisin -> käin (Past -> Pres)
R:VERB:FORM:SPELL:CASE	Verb form, spelling and capitalization error	jstume istume -> Istusime (Pres -> Past)
R:LEX:SPELL	Word choice and spelling error	laksin läksin -> käisin
R:LEX:CASE	Word choice and capitalization error	võimalikult -> Võimalik (ADV -> ADJ)
R:LEX:NOM:FORM	Word choice and nominal form error	muusikaid -> muusikastiilid (Par -> Nom)
R:LEX:VERB:FORM	Word choice and verb form error	(mina) oli -> (mina) käisin (3rd person -> 1st person)
R:LEX:WO	Word choice error affects word order	läbi interneti -> interneti kaudu
R:LEX:WS	Word choice and whitespace error	oma teist -> teineteist
R:WO:NOM:FORM	Word order error affects the choice of nominal form	pealinn Islandil -> Islandi pealinn (Ade -> Gen)

II. Prompt in English

Modify the input text by generating errors that a learner of the Estonian language might make. As the output text, return the input text into which you have generated errors. Generate spelling, grammar, word choice, word order, punctuation, and style errors in the input text. If there are errors in the input text, do not correct them, but generate additional errors. There are four examples for the task: Input Text: `{correct}` Output Text: `{incorrect}` Input Text: `{correct}` Output Text: `{incorrect}` Input Text: `{correct}` Output Text: `{incorrect}` Input Text: `{correct}` Output Text: `{incorrect}` Input Text: `{input}` Output Text:

Table 14. 4-shot prompt template in English.

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Martin Vainikko**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Estonian Synthetic Error Generation by Prompting for Grammatical Error Correction,
   *(*title of thesis)

   supervised by Agnes Luhtaru and Mark Fišel.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Martin Vainikko
20/05/2024