Multilingual Grammatical Error Detection And Its Applications to Prompt-Based Correction

by

Gustavo Sutter Pessurno de Carvalho

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2023

Abstract

Grammatical Error Correction (GEC) and Grammatical Error Correction (GED) are two important tasks in the study of writing assistant technologies. Given an input sentence, the former aims to output a corrected version of the sentence, while the latter's goal is to indicate in which words of the sentence errors occur. Both tasks are relevant for real-world applications that help native speakers and language learners to write better. Naturally, these two areas have attracted the attention of the research community and have been studied in the context of modern neural networks. This work focuses on the study of multilingual GED models and how they can be used to improve GEC performed by large language models (LLMs).

We study the difference in performance between GED models trained in a single language and models that undergo multilingual training. We expand the list of datasets used for multilingual GED to further experiment with cross-dataset and cross-lingual generalization of detection models. Our results go against previous findings and indicate that multilingual GED models are as good as monolingual ones when evaluated in the in-domain languages. Furthermore, multilingual models show better generalization to novel languages seen only at test time.

Making use of the GED models we study, we propose two methods to improve corrections of prompt-based GEC using LLMs. The first method aims to mitigate overcorrection by using a detection model to determine if a sentence has any mistakes before feeding it to the LLM. The second method uses the sequence of GED tags to select the in-context examples provided in the prompt. We perform experiments in English, Czech, German and Russian, using Llama2 and GPT3.5. The results show that both methods increase the performance of prompt-based GEC and point to a promising direction of using GED models as part of the correction pipeline performed by LLMs.

Acknowledgements

First, I would like to thank my supervisor, Professor Pascal Poupart, for all the support during my Master's degree. His valuable guidance and vast knowledge were vital for me throughout the degree.

I would like to thank my family, every single one of them always motivated me to pursue my dreams and always believed in me. For my parents, Andre and Maria Isabel, I would like to express how thankful I am for you always incentivizing me to learn and explore. You are the best parents I could have ever wished for. I love you.

Moving to a different country without knowing anyone during the lockdown period was the most challenging moment of my life. I would like to thank my friends Fernando, Luiz Guilherme, Matheus Gomes, Matheus Tassi, Philippe, Victor and Vittorio, who kept me company online during this hard transition period. I would also like to thank Carol, Julia, Juliana and Vrech, for accepting me into their friend group and making me feel welcome in Canada. Finally, I want to thank my roommates João Paulo and Matheus, who are an important part of my daily life.

Last, but certainly not least, I would like to thank my girlfriend Shelby. I want to thank you for always being there to listen and help, especially in moments when anxiety has taken control over me. I also want to thank you for all the fun moments, every meal we cooked and every show we watched were important and made me feel loved in this once-unknown city. You make me the happiest person on earth.

Dedication

This is dedicated to my parents and my brother.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgements iv

Dedication v

List of Figures ix

List of Tables x

1 Introduction 1

1.1 Motivation 1

1.2 Tasks 3

1.3 Research Questions 3

1.4 Contributions 4

1.5 Thesis Outline 5

2 Background 6

2.1 Neural Networks 6

2.2 Sequential Data and Recurrent Neural Networks 7

2.3 Transformers . . . 9
2.3.1 Decoder-Only Models . . . 13
2.3.2 BERT - Encoder-Only Models . . . 14
2.4 NLP Beyond English . . . 15
2.5 Grammatical Error Correction . . . 16
2.5.1 Datasets . . . 17
2.5.2 Evaluation Metrics . . . 20
2.6 Grammatical Error Detection . . . 22
2.6.1 Datasets . . . 23

3 Related Work 25
3.1 Grammatical Error Detection . . . 25
3.2 Grammatical Error Correction . . . 26
3.2.1 Supervised Methods . . . 26
3.2.2 Unsupervised Methods . . . 27

4 Multilingual Grammatical Error Detection 28
4.1 Methodology . . . 30
4.2 Experimental Setup . . . 31
4.3 Results and Analysis . . . 31
4.3.1 In-domain . . . 31
4.3.2 Cross-dataset . . . 32
4.3.3 Cross-Lingual . . . 32

5 Detection Assisted Prompt-Based Grammatical Error Correction 34
5.1 Methodology . . . 35
5.1.1 GED Filtering of Correct Senteces . . . 35
5.1.2 GED for In-context Example Selection . . . 36

5.2 Experimental Setup 37
5.3 Results and Analysis 39
5.3.1 GED Filter 39
5.3.2 TF-IDF GED 42
5.3.3 Combining the Methods 43

6 Conclusions **45**

References **47**

List of Figures

1.1 Example of GEC and GED tasks. On the former, a corrected sentence is returned as output. On the latter, a correct (`C`) or incorrect (`I`) label is produced for each word. 1

2.1 Illustration of a feedforward neural network with a single hidden layer. . . 7

2.2 Visualization of the different types of sequential data tasks. 8

2.3 Schematic representation of an RNN unrolled over the input steps. 9

2.4 The Transformer architecture from Vaswani et al. [71] 12

4.1 Difference between a collection of models trained on monolingual data (left) and a single model trained with multilingual data (right). 29

4.2 Results of cross-lingual transfer across all models. 33

5.1 Illustration of the TF-IDF GED framework proposed for in-context example selection. First, the sentences are passed to a GED model to extract the detection tags, and then TF-IDF is used on the tags to extract feature vectors. Once with the feature vectors, we compute the similarity between the test sentence and the training examples to obtain the in-context examples. 36

5.2 Effect of filtering out correct sentences on the W&I+LOCNESS development set. 39

5.3 Comparison between the random selection, SBERT similarity, and TF-IDF GED on the W&I+LOCNESS development set. 42

5.4 Comparison between the random selection, and TF-IDF GED with multiple levels of granularity and ground truth data on the W&I+LOCNESS development set. 43

List of Tables

2.1 Example of error types in GEC from [11] 16

2.2 Overall statistics of the different GEC datasets from [11]. A question mark (?) indicates unknown or approximated information. Notice that the Arabic dataset is split into documents instead of sentences. 18

2.3 Example of different levels of granularity in GED. 22

2.4 Statitics of the datasets from MultiGED-2023. Notice that English-REALEC only has dev and test splits. 23

4.1 Statistic of the new GED datasets used for evaluation. Note that the Arabic dataset is split into documents rather than sentences. 31

4.2 Results of monolingual and multilingual models on the in-domain datasets. 32

4.3 Results of monolingual and multilingual models on the cross-dataset evaluation. 33

5.1 Results of the multilingual GED model when evaluated on a sentence level. For this experiment, the sentence is considered incorrect if any of its words are classified as incorrect. 36

5.2 Complete results on the W&I+LOCNESS development set. Checkmarks (✓) indicated if the corresponding method was used. The reported scores are the average between five random seeds, with indicated standard deviation. Bold values indicate the highest score for each number of in-context examples. 40

5.3 Results of both methods on the non-English datasets. Checkmarks (✓) indicated if the corresponding method was used. The reported scores are the average between three random seeds, with indicated standard deviation. Bold values indicate the highest score each number of in-context examples. 41

5.4 Results of the combination of filtering and TF-IDF GED in test sets in English, Czech, German and Russian. Dashes (-) indicate that the method was not available in the language. Models are divided into supervised and unsupervised, the first being supervised and the last unsupervised. 44

Chapter 1

Introduction

1.1 Motivation

The advances in machine learning driven by the deep learning revolution in the early 2010s have drastically impacted natural language processing (NLP). From speech recognition, machine translation, and natural language generation to many others, the use of neural networks boosted the performance of ML systems by large margins. Not differently, areas related to writing assistance have also been explored and greatly benefit from modern approaches.

Mainly, two tasks have been studied in this context: Grammatical Error Detection (GED) and Grammatical Error Correction (GEC). The former is interested in detecting which words in a sentence are incorrect and the latter focuses on generating a correction.

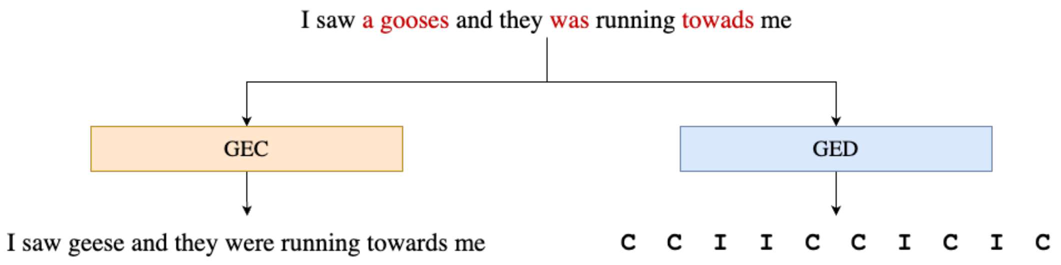
I saw a gooses and they was running towads me
GEC
GED
I saw geese and they were running towards me
C C I I C C I C I C

Figure 1.1: Example of GEC and GED tasks. On the former, a corrected sentence is returned as output. On the latter, a correct (`C`) or incorrect (`I`) label is produced for each word.

An example of both tasks is provided in Figure 1.1. These systems are present in general public applications, such as Microsoft Editor [1] and Grammarly [2], and as part of toolboxes for proofreading professionals and language learning instructors. Detection can be used as an explicit step toward correction but also developed as a final goal.

Analyzing the example provided in Figure 1.1 we can see some of the error types covered in the tasks. More specifically the extra *a* is a determiner error, the wrong pluralization *gooses* is a morphology error, *was* in place of *were* is a verb agreement error and *towads* is a spelling error. From this example, it is also possible to notice that to perform detection and correction both local and global sentence structures should be taken into consideration.

Although English is the language with the largest number of speakers in the world, more than three-fourths of the world's population does not speak English, most of which are in developing countries. Therefore, to make NLP accessible we should go beyond English. In addition, studying different languages gives rise to new challenges, such as lower data availability and more complex syntax and morphology structures. This motivates the development of more powerful and data-efficient algorithms. The study of multilingual models that can improve the performance of low-resource languages by generalizing from high-resource ones has been demonstrated to be an effective solution.

Most of the work in GED and GEC was developed in English, with some efforts being developed in other languages. One instance is the MultiGED-2023 shared task [72], which released new datasets to promote the development of multilingual GED. With the released datasets and the submissions to the task, results indicated that monolingual models are still better for detection than a multilingual solution that can be shared across languages.

Back to the general problem of GEC, recent works have investigated the use of Large Language Models (LLMs) on the task. The idea is to prompt the model by describing the task and providing some in-context examples. This approach allows very quick implementation of GEC systems, as a variety of LLMs are available through API calls and require no training. However, the overall performance of prompt-based GEC is still far from the results achieved by supervised methods. Among the reasons for this difference, is LLMs' tendency to overcorrect and generate corrections that are not expected by the test dataset. This gap motivates more work studying ways in which prompt-based GEC can get closer to supervised models.

[1]https://www.microsoft.com/en-ca/microsoft-365/microsoft-editor
[2]https://app.grammarly.com/

1.2 Tasks

Definition 1 *Grammatical Error Correction (GEC) is a task in which an input sentence, potentially containing grammatical errors, is presented, and the goal is to generate a corrected version as the output. GEC involves grammar, misspellings, punctuation, word choice and fluency. It often requires an understanding of the intended meaning of the writer in order to perform the correction. The output should always preserve this meaning, and no unnecessary changes should be made to the input.*

In practice, GEC can be performed in both a supervised and unsupervised manner. Supervised methods make use of a paired dataset to learn the mapping between input sentences and their correction. On the other hand, unsupervised methods often rely on language models, which can be done through prompting or methods that use the probability of a sentence.

Definition 2 *Grammatical Error Detection (GED) is a task focused on identifying the errors within a given sentence. Typically framed as a word-level classification task, GED aims to predict which words in a sentence are incorrect. GED can adopt binary or multiclass approaches, the former indicating correctness and the latter dealing with specific error types.*

1.3 Research Questions

In this thesis we seek answers to the following research questions:

Question 1 *Grammatical rules are language-specific, however, language groups tend to share different grammatical features. This leads to a natural trade-off between focusing on a single language at a time and trying to exploit shared features between languages. Can a single model learn to perform GED in multiple languages at the same level as models trained on monolingual datasets?*

Question 2 *GED models are usually fine-tuned on top of massively multilingual foundation models. Given the fact that languages share grammatical and structural similarities, models should be able to perform zero-shot detection in languages not seen during GED training. How well do monolingual and multilingual GED models perform in zero-shot cross-lingual evaluation?*

Question 3 *A notable problem with prompt-based GEC with LLM is overcorrection. We are interested in making use of a GED system to improve corrections by avoiding that sentences without errors are changed by the LLM. What is the impact of using a GED model to filter out correct sentences before performing prompt-based GEC?*

Question 4 *Semantical similarity is often used to determine relevant in-context examples from the training set, however, providing sentences with similar meaning is not useful for GEC. Instead, we hypothesize that computing the similarity based on GED tags may allow more relevant examples to be retrieved. What method is better for GEC in-context example selection: semantical similarity or detection tags similarity?*

1.4 Contributions

On the GED side, we perform various experiments comparing monolingual and multilingual models. Evaluation is performed with a focus on in-domain, cross-dataset and cross-lingual generalization. We make sure to use reasonable choices of hyperparameters and multiple random seeds to capture the variance in the results. We show that contrary to what was stated in the MultiGED task report, models trained on a single language and models trained on multilingual corpora obtain equivalent performance. Both in-domain and cross-dataset evaluation results show that the strategies achieve very close results.

For cross-lingual generalization of GED models, we experiment with four extra new languages covering more diverse grammatical rules. We show that the multilingual model can generalize to novel languages and achieve reasonable results without ever being trained to perform detection in the language. We also observe that transfer is not equal to all languages and the model can struggle to generalize to certain languages.

Regarding prompt-based GEC, we demonstrate two ways in which GED can be used to improve performance in English, Czech, German and Russian. First, we use the aforementioned multilingual GED model to filter out sentences that are already correct. This improves the scores of the GEC models and reduces the number of sentences fed to the LLM, most likely the most costly component of the pipeline.

Secondly, we show that detection tags can be used for in-context example selection. We demonstrate that features extracted from predicted GED tags improve the corrections. Considering that GED systems can be trained with relatively small datasets and transfer well between datasets and certain languages, it is an effective way to improve GEC performed by LLM.

To the best of our knowledge, we are the first to benchmark Llama2 [69] on the GEC task. For the other languages, we use GPT3.5, and we are the first work to explore Russian and Czech GEC through prompting LLMs.

1.5 Thesis Outline

This thesis is organized as follows.

- In Chapter 2, we introduce fundamental topics in modern NLP, such as Feedforward Neural Networks, Recurrent Neural Networks and Transformers. As well as background information on Multilingual NLP, Grammatical Error Correction and Grammatical Error Detection.
- In Chapter 3, we describe related work in GED and GEC. This situates the methods we study in the overall landscape of the fields.
- In Chapter 4, we study the differences between monolingual and multilingual models under the lens of in-domain, cross-dataset and cross-lingual evaluation.
- In Chapter 5, we propose two methods that make use of GED models to improve prompt-based GEC with LLMs.
- In Chapter 6, we summarize our work, discussing limitations and indicating future directions.

Chapter 2

Background

In this chapter, we cover the background work that is used as a basis throughout the thesis. First, we introduce neural networks and the foundation of deep learning models. Second, we cover the paradigm of sequential data, exploring its formulation and recurrent neural architectures. Next, we explore the Transformer, the prevalent model in modern natural language processing, and its variations. Next, we discuss the importance of expanding NLP research beyond English. Finally, we introduce Grammatical Error Correction and Grammatical Error Detection, defining the problems, and going through popular datasets and evaluation metrics.

2.1 Neural Networks

Artificial Neural Networks, or simply Neural Networks, are a type of machine learning model loosely inspired by the brain. They consist of interconnected nodes, called artificial neurons, that are organized in layers. Each neuron takes as input the signals from the previous layer and weights them to compute its output, referred to as activation. The number of layers between the input and output layers and the number of neurons in each layer are both model hyperparameters.

A fundamental type of neural network is the feed-forward neural network, in which all neurons in a layer have a weighted connection to all neurons in the previous layer. Mathematically, a layer can be defined as a function $f(\boldsymbol{x}; W, \boldsymbol{b}) = \sigma(\boldsymbol{x}W + \boldsymbol{b})$, where $\boldsymbol{x}$ is an input vector, W is a weight matrix, $\boldsymbol{b}$ is bias vector, and σ a non-linear function referred

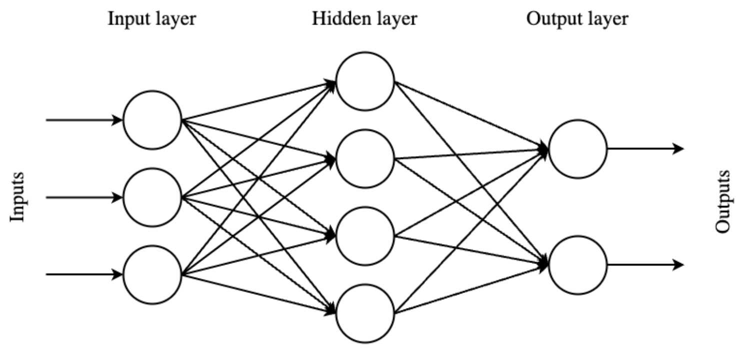

Figure 2.1: Illustration of a feedforward neural network with a single hidden layer.

to as activation function. The activation is used so that neural networks can learn non-linear functions and, at the same time, make neural networks a non-convex optimization problem.

With the use of deeper models, we get powerful models that are able to learn complex relationships between features and targets. To support bigger neural networks while keeping training times within a reasonable length, heavy parallelism is exploited using GPUs and TPUs.

To train the network the input is passed through all layers and the final output is compared to the true values/labels using a loss function. Loss functions are differentiable functions that measure how good/bad the neural network is, common loss functions used for classification and regression tasks are Cross-Entropy and Mean Squared Error, respectively. Using backpropagation to compute their derivatives, the parameters are updated using Stochastic Gradient Descent or its variations.

2.2 Sequential Data and Recurrent Neural Networks

In a wide variety of applications, we are interested in data that is structured as sequences of potentially varying lengths. As depicted in Figure 2.2, this includes sequential input with single output (many-to-one), single input with sequential output (one-to-many), and sequential input and sequential output (many-to-many). Some examples of each paradigm are sentiment analysis, image captioning and machine translation, respectively.

Further exploring the many-to-many setting, it can be broken down into the case in which input and output have the same length and the case in which they do not. The case where the lengths differ the problem is usually referred to as sequence-to-sequence. To illustrate the difference between the two, we can look into part-of-speech tagging and machine translation. In the former, each word in the input sequence is mapped to one tag, but in the latter, there's no one-to-one mapping between words, which can result in different lengths.

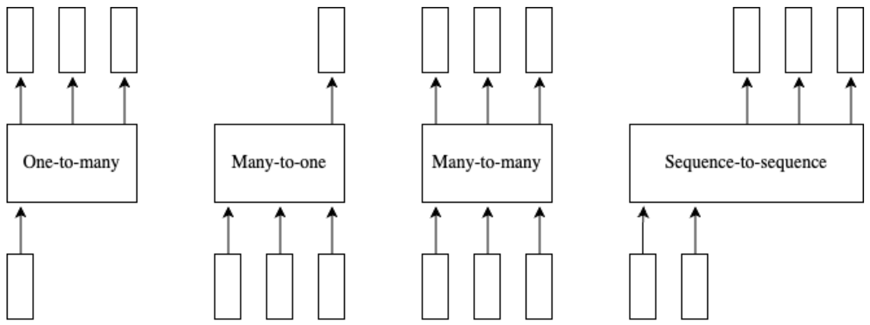

Figure 2.2: Visualization of the different types of sequential data tasks.

As simple feedforward neural networks are not ideal for dealing with sequential data, specific architectures were developed to address this problem. More notably Recurrent Neural Networks (RNNs) were the prevalent way to deal with sequential data for many years and are still widely used. In their basic configuration, RNNs keep a hidden state vector that is passed along as the sequence is being processed, capturing the dependency on previous steps. As demonstrated in Figure 2.3, at each step the previous hidden state $\boldsymbol{h}_{t-1}$ and the current input $\boldsymbol{x}_t$ are used to compute the current hidden state $\boldsymbol{h}_t$ and the current output $\hat{\boldsymbol{y}}_t$ according to following equations:

$$\begin{aligned}\boldsymbol{h}_t &= g(\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{hx}\boldsymbol{x}_t)\\ \hat{\boldsymbol{y}}_t &= f(\boldsymbol{W}_y\boldsymbol{h}_t)\end{aligned}$$

where f,g are activation functions, and $\boldsymbol{W}_{hh}$, $\boldsymbol{W}_{hx}$, $\boldsymbol{W}_y$ are learnable weight matrices.

Due to its form, some optimization problems arise with long sequences, resulting in exploding/vanishing gradients on the earlier steps. To address this problem, more complex RNN architectures such as Long Short Term Memory (LSTM) [28] and Gated Recurrent Unit (GRU) [12] use gating mechanisms when processing the hidden state at each step.

For sequence-to-sequence problems, it was proposed to use one RNN to read the input sequence and another RNN to generate the output sequence. In this encoder-decoder

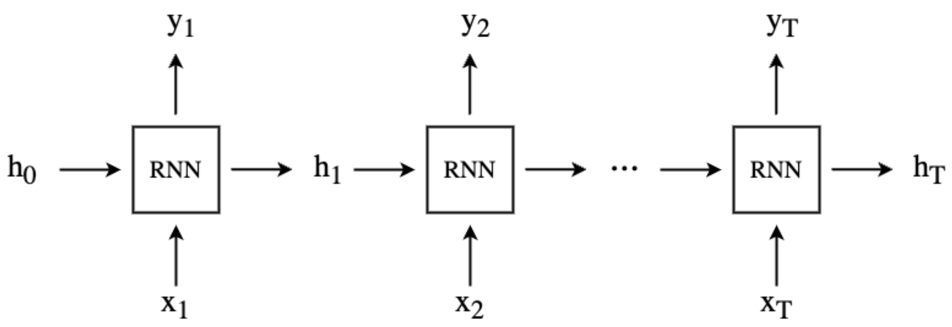

Figure 2.3: Schematic representation of an RNN unrolled over the input steps.

architecture, the input sequence is processed by the encoder network producing a fixed-length context vector (often the last hidden state). The decoder generates each element of the output sequence conditioned on the previous outputs and the context vector generated by the encoder. The whole system, both encoder and decoder, is trained end-to-end using backpropagation.

An extension to the architecture just described is to use a different context vector at each step instead of the same context vector every time. This mechanism, called attention, allows the decoder to select which parts of the input are more relevant at each decoding step. Concretely, the context at a certain step is computed as a convex combination of the hidden states of the encoder, where the coefficients are computed using a simple feed-forward network that receives as input the current hidden state of the decoder and the corresponding hidden state of the encoder. As all modules are still differentiable the system is trained end-to-end using backpropagation.

2.3 Transformers

The main bottleneck in RNNs is their recurrent nature, as the sequence has to be processed element by element there is a limit to the amount of parallelism that can be exploited. Because of that, training time becomes prohibitive when dealing with longer sequences or massive datasets. The Transformer architecture solves this problem by avoiding recurrence and relying completely on the attention mechanism. The resulting architecture is extremely parallelizable, allowing it to be used with massive datasets and obtain the state-of-the-art in a wide variety of deep learning tasks.

Following the general framework for sequence-to-sequence learning, the Transformer

[71] is an encoder-decoder architecture. The encoder maps the sequence of input elements $(\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$ into a sequence of fixed-length representations $(\boldsymbol{z}_1, ..., \boldsymbol{z}_n)$ that is given to the decoder to generate the output sequence $(\boldsymbol{y}_1, ..., \boldsymbol{y}_m)$. A complete picture of the Transformer is provided in Figure 2.4. Before defining the architecture of the encoder and decoder layers let us go through the relevant components.

Scaled Dot-Product Attention: The inputs to this module are queries and keys of dimension d_k and values of dimension d_v. To compute the output, the dot product between queries and keys is divided by $\sqrt{d_k}$ and passed through a softmax to get the weights used to combine the values. Formally, considering the matrices of packed keys, queries and values as K, V, and Q, respectively, the matrix of outputs is given by:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Multi-head Attention: The main idea of this module is to perform attention with lower dimension projections of the queries, keys and values, and then concatenate the outputs. The idea is that the independent attention outputs can capture different aspects of the sequence. The Multi-head Attention can be computed as:

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned}$$

where W^Q, W^K, W^V W_O are learnable projection matrices. When the same set of vectors is passed for keys, vectors and queries it is denominated self-attention.

Position-wise Feed-Forward Networks: This module receives a sequence of vectors and processes each separately by the same feed-forward network. The network consists of a single hidden layer with ReLU activation and higher dimensionality than the input vectors. Considering each vector as x they are processed as follows:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

With all the components defined, we can build the encoder and decoder layers.

Each encoder layer receives a set of vectors, each with dimension d_{model}, and has two sub-layers: multi-head self-attention followed by a position-wise feed-forward network. In both sub-layers, there are residual connections followed by layer normalization modules.

The input and output of both sub-layers and by consequence of the whole layer, have the same dimensionality d_{model}.

The decoder layer has an extra sub-layer module, which performs multi-head attention over the output of the encoder. This "encoder-decoder attention" is positioned in between the two sub-layers described on the encoder and also has a residual connection and layer normalization. Another difference is that the decoder's self-attention is altered so that each position cannot attend to subsequent positions. This is often referred to as masked self-attention and allows the decoder to work in parallel while preserving the auto-regressive property of the decoding process.

One important property of both multi-head attention and position-wise feed-forward is permutation equivariance, that is, permuting the input vectors would simply permute the output vector, without changing the actual values. This property is not interesting for NLP, as the order in which tokens appear in a sentence is very relevant. To ensure that the order in the sequence is captured by the model a positional encoding is added to each input embedding.

Since their introduction, Transformers have been widely used in NLP and even in other areas of machine learning. The highly parallelizable architecture allows training with large datasets. Because of this capability, a new paradigm became the norm for NLP: unsupervised pre-training with massive data and task-specific fine-tuning with smaller supervised datasets. Among the most relevant encoder-decoder Transformer architectures are BART [38] and T5 [54].

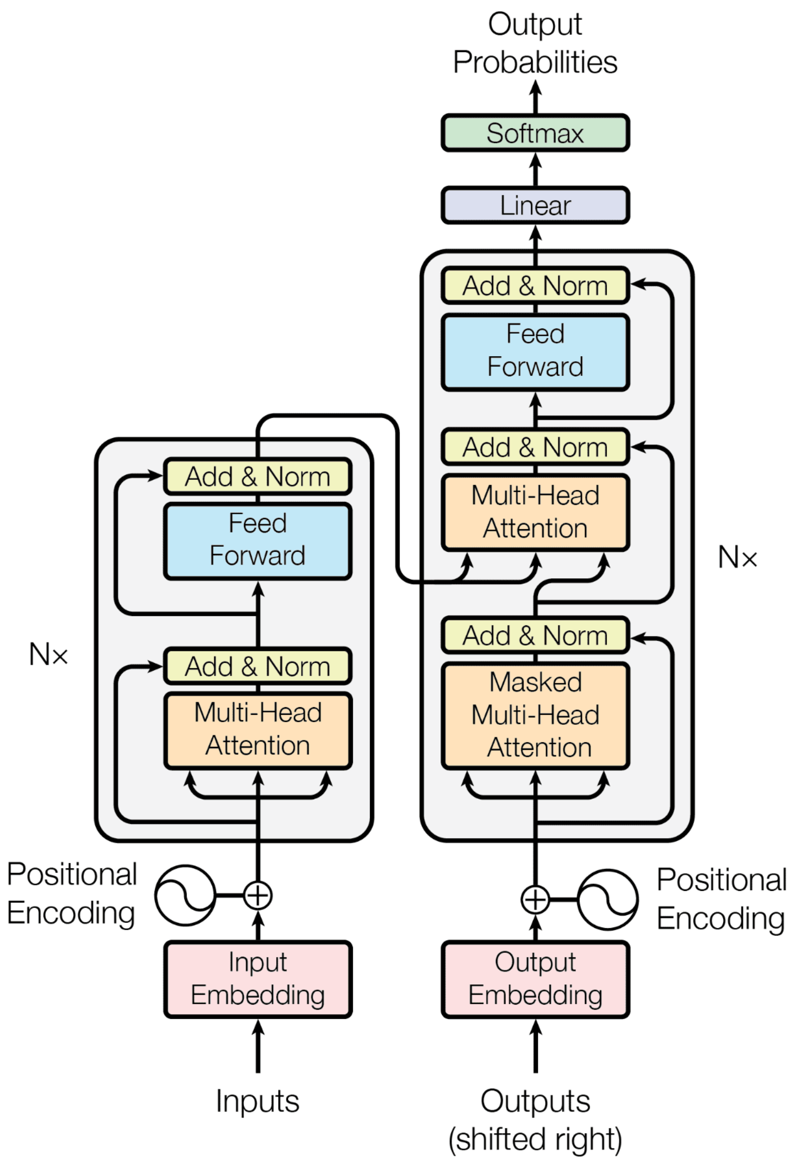

Figure 2.4: The Transformer architecture from Vaswani et al. [71]

2.3.1 Decoder-Only Models

The Generative Pre-trained Transformer [52] (GPT) family is a series of decoder-only models that work as language models. Language models are probabilistic models that model the distribution of a series of words based on some training corpus. More specifically, GPTs are autoregressive language models, as they generate the next word given the words seen up to that point in the sequence, that is $\Pr(y_t|y_1, ..., y_{t-1})$. Such models are trained by maximizing the likelihood of the sequences of words in the training corpus.

Being a decoder-only model means that the layers in the GPT model have a masked multi-head attention sub-layer and a position-wise feed-forward sub-layer, both wrapped with residual connections and layer normalization. Notice that there is no cross-attention module as there is no encoder output to attend to.

The first paper in the GPT series introduced the idea of pre-training Transformers using massive quantities of unsupervised datasets. As it is an autoregressive language model, GPT can be trained using next-word prediction, requiring no human-annotated data. Therefore it can be pre-trained using large amounts of raw text that is widely available on the internet. The main idea is that during unsupervised pre-training the model can learn general language information that helps to solve the downstream tasks. Starting from the pre-trained model fine-tuning on a specific task is performed by adding a new layer on top of the final layer.

Later, GPT-2 [53] showed that by using a larger language model it is possible to obtain great performance on multiple tasks in a zero-shot setting. In this model, the whole task (description, input and output) is provided to the model using the natural language as the interface, which is referred to as prompting. Because of that, fine-tuning does not require new layers on top of the network, it is performed using the same objective function as pre-training. The main reason the model is able to obtain this zero-shot performance is its scale. GPT-2 is a 1.5 billion parameter model, making it more than 10 times bigger than the original GPT. In addition to that, GPT-2 is pre-trained with 40GB of text data from over 8 million documents.

The zero-shot capabilities of GPT-2 opened a new era in NLP in which by scaling the model and the amount of data we can perform tasks with only unsupervised training. Its successor, GPT-3 [7] shows that with a 175B model (100 times bigger than GPT-2) and an even bigger dataset, it is possible to perform a wide variety of tasks using only unsupervised pre-training. The large scale lead such model to be referred to as large language models (LLMs). They demonstrate that by prompting the model with a task description and a very small number of examples it is able to perform an enormous number of tasks successfully.

Following that, the FLAN [75] model shows that fine-tuning a collection of datasets via instructions makes the model better on unseen tasks. This method, called instruction tuning became very present as part of large language model training.

As of today, such models have been scaled even more. The latest model of the GPT family, GPT-4 [51], has demonstrated human-level performance in various professional and academic benchmarks. The model details however are not released to the public and access to it is only available through an API. Alternatively, the Llama family [68] [69] are open-source LLMs that were released with multiple model sizes and versions fine-tuned on instruction following datasets. In addition to the Llama model other models were released to the public, for example, Falcon [2] and GPT-J [74].

2.3.2 BERT - Encoder-Only Models

While decoder-only models focus on the generation task in an autoregressive way, encoder-only models are used to perform sentence-level or token-level representation learning. Such models can be used to perform classification or regression tasks by observing the whole sentence, which is often referred to as bidirectional attention in contrast with the unidirectional attention present in decoder-only models. Encoder-only models take a sequence of tokens as input and in a single step return a sequence of the same size as output. In order to perform different tasks, special tokens can be added to represent for example separators or sequence-level classification tokens.

The first model following this scheme was the Bidirectional Encoder Representations from Transformers (BERT) [21], which demonstrated excellent performance by using unsupervised pre-training followed by task-specific supervised fine-tuning. The model is pre-trained in two ways: masked language modelling (MLM) and next-sentence prediction (NSP). In masked language modelling, tokens are substituted by a mask randomly, and the model should predict the true value of the masked tokens. In next-sentence prediction, that model is given two sentences and the model should predict if they follow each other or if the second sentence was randomly selected from the corpus. Notice that both tasks only require a corpus with raw text, without any annotations.

Starting from a pre-trained BERT it can be used to perform any sentence-level or token-level tasks by adding a new layer on top of the model and fine-tuning the weights using a supervised dataset. Because of its encoder-only structure, BERT is very efficient for training and especially inference, as it does not perform autoregressive decoding. In its original paper, BERT has been applied successfully to various tasks such as sentiment analysis, semantic similarity, question answering, and many more. Originally BERT was proposed with two configuration sizes, $\text{BERT}_{\text{Base}}$ with 110M parameters and $\text{BERT}_{\text{Large}}$ with 340M parameters.

Various works built on top of BERT to make it better or more efficient. RoBERTa [41] is a variation of BERT that is pre-trained only using MLM and different hyperparameters such as batch size and learning rate. ELECTRA changes the pre-training strategy by having a generator that performs MLM and a discriminator that tries to identify which tokens were replaced by the generator. From an efficiency perspective, DistillBERT [62] makes use of knowledge distillation to create a model that is 60% faster than BERT while preserving 95% of its performance.

2.4 NLP Beyond English

The vast majority of NLP research is done in English, according to recent estimates, almost 70% of the papers evaluate their method only in English [61]. Considering that around 400 languages have more than 1M speakers, it is clear that to make AI more accessible the diversity of languages should be increased. In addition to the accessibility perspective, going beyond English also allows us to evaluate how algorithms perform in more complex situations such as richer morphology or more diverse scripts.

Being aware of this problem the research community has made some efforts to collect datasets in different languages. This is a hard and costly task, as the collection of labelled datasets for a variety of tasks requires qualified annotators. Because of that, even when considering the existing multilingual tasks, there is still a very high bias towards languages spoken in richer countries. Take as an example XTREME, the main multilingual benchmark, although it covers up to 40 languages in 9 different tasks, only 3 of the tasks are available in more than 15 languages.

When it comes to modern language models, we have seen efforts to make multilingual versions of the widely available pre-trained models, for example, mBERT [21], mT5[79] and XLM-RoBERTa[16]. As pre-training data does not require annotations, the amount of languages that can be covered is higher. Nevertheless, it is an active challenge to collect corpora where more languages are well represented. From a performance point of view, these models tend to perform as well as their monolingual counterparts when finetuned on a specific task and often exhibit good zero-shot cross-lingual transferability. However, as the number of languages increases the overall performance drops for all languages, a phenomenon that is known as the curse of multilinguality.

Type	**Error**	**Correction**
Preposition	I sat in the talk	I sat in on the talk
Morphology	dreamed	dreamt
Determiner	I like the ice cream	I like ice cream
Tense/Aspect	I like kiss you	I like kissing you
Agreement	She likes him and kiss him	She likes him and kisses him
Syntax	I have not the book	I do not have the book
Punctuation	We met they talked and left	We met, they talked and left
Unidiomatic	We had a big conversation	We had a long conversation
Multiple	I sea the see from the seasoar	I saw the sea from the seesaw

Table 2.1: Example of error types in GEC from [11]

2.5 Grammatical Error Correction

Grammatical Error Correction (GEC) is an NLP task in which a potentially incorrect sentence is given as input and its correction should be generated as output. It has applications that benefit both native and non-native writers, and it is present in popular writing tools.

Even though this definition looks simple at first, it hides a series of nuances that make this problem extremely tricky to solve. First, we are interested in more than only grammatical errors, as GEC commonly also covers misspelling, word choice and punctuation. However, more complex text editing tasks such as factual corrections or writing style are usually not considered part of GEC. A small sample of the types of errors that we are interested in is provided in Table 2.1

Another aspect of the problem definition is that the same input sentence may have a set of possible corrections. Therefore evaluating corrections, especially in an automated fashion, is hard and requires a certain level of compromise. Some corpora provided multiple annotation for the same sentence to account for alternative solutions. In practice, the field tends to adopt as a guideline to have minimal modifications of the original text while preserving its meaning. Different corpora may focus more on the principle of minimal correction while others may focus on fluency.

Finally, GEC is often framed as a sentence-level task, which makes it sometimes impossible to capture the whole intent of the writer. Because of that, there is an unavoidable noise that comes from the fact that a correct sentence in a broader context may look incorrect when observed in isolation. This is amplified by the fact that corpora are usually annotated having the whole context available but GEC itself is performed in isolated sentences in datasets and shared tasks.

2.5.1 Datasets

The construction of GEC datasets relies on paired data where each sentence is accompanied by its correction. Collecting such datasets is labour-intensive and ideally requires annotators with higher education in the area, such as language instructors or teachers. In general, the sentences can come from native speakers (L1) or second language learners (L2), which changes the distribution of errors observed. More specifically, language learners are usually broken on a six-level scale (A1, A2, B1, B2, C1, C2) [1] and can come from different native languages, which also adds more diversity to the errors.

We will now introduce some of the most popular GEC datasets used by the research community. This list does not intend to be exhaustive, as we focus on the datasets that are more relevant to our work. A summary of all the datasets listed is provided on Table 2.2.

English Datasets

FCE: The First Certificate in English (FCE) corpus [81] is a subset of the Cambridge Learner Corpus [48] that consists of 1244 scripts ($\sim$ 531k words) written by international language learners (B1-B2 level). Each script contains two answers to an exam question and can be short essays, letters or descriptions. The dataset has 71 error types manually annotated and is divided into train, development and test. FCE was part of the HOO-2012 [20] and BEA-2019 [9] shared tasks.

NUCLE: The National University of Singapore Corpus of Learner English (NUCLE) [19] is comprised of argumentative essays written by undergraduate students from Singapore (approximately C1 level). It consists of 1397 essays ($\sim$1.16M words), annotated with 28 different error types. The dataset was used as the official training set for the CoNLL-2013 and CoNLL2014 shared tasks [46] [47], and as one of the training sets on the BEA-2019 shared task.

Lang-8: The Lang-8 Corpus of Learner English [67] was extracted from the Lang-8 [2] online community for language learners. The dataset consists of 100,000 submissions

[1]https://www.coe.int/en/web/common-european-framework-reference-languages/level-descriptions

[2]https://lang-8.com/

Corpus	**Language**	**Use**	**Nr. Senteces**	**Nr. Tokens**	**Nr. Refs**	**Nr. Error Types**	**Level**	**Domain**
FCE	English	Train	28.3k	454k	1	71	B1-B2	Exams
		Dev	2.2k	34.7k	1	71	B1-B2	Exams
		Test	2.7k	41.9k	1	71	B1-B2	Exams
NUCLE	English	Train	57.1k	1.16M	1	28	C1	Essays
Lang-8	English	Train	1.03M	11.8M	1-8	0	A1-C2?	Web
W&I+LOCNESS	English	Train	3k	628k	1	55	A1-C2	Exams
		Dev	4.4k	87.0k	1	55	A1-Native	Exams, Essays
		Test	4.5k	85.7k	5	55	A1-Native	Exams, Essays
QALB-2014	Arabic	Train	19.4k	1M	1	7	Native	Web
		Dev	1k	53.8k	1	7	Native	Web
		Test	948	51.3k	1	7	Native	Web
QALB-2015	Arabic	Train	310	43.3k	1	7	A1-C2	Essays
		Dev	154	24.7k	1	7	A1-C2	Essays
		Test	158	22.8k	1	7	A1-C2	Essays
		Test	920	48.5k	1	7	Native	Web
NLPCC-2018	Chinese	Train	717k	14.1M	1-21	0	A1-C2?	Web
		Test	2k	61.3k	1-2	4	A1-C2?	Essays
AKCES-GEC	Czech	Train	42.2k	447k	1	25	A1-Native	Essays, Exams
		Dev	2.5k	28.0k	2	25	A1-Native	Essays, Exams
		Test	2.7k	30.4k	2	25	A1-Native	Essays, Exams
GECCC	Czech	Train	66.6k	750k	1	65	A1-Native	Essays, Exams, Web
		Dev	8.5k	101k	1-2	65	A1-Native	Essays, Exams, Web
		Test	7.9k	98.1k	2	65	A1-Native	Essays, Exams, Web
Falko-MERLIN	German	Train	19.2k	305k	1	56	A1-C2	Essays, Exams
		Dev	2.5k	39.5k	1	56	A1-C2	Essays, Exams
		Test	2.3k	36.6k	1	56	A1-C2	Essays, Exams
RULEC-GEC	Russian	Train	5k	83.4k	1	23	C1-C2	Essays
		Dev	2.5k	41.2k	1	23	C1-C2	Essays
		Test	5k	81.7k	1	23	C1-C2	Essays
Uz & Eryiğit [70]	Turkish	Test	106	823	1	23	?	Papers

Table 2.2: Overall statistics of the different GEC datasets from [11]. A question mark (?) indicates unknown or approximated information. Notice that the Arabic dataset is split into documents instead of sentences.

(∼11.8M words) from all proficiency levels (A1-C2). Even though the dataset is one the largest available, its corrections are considered noisy as they are not provided by professional annotators. The Lang-8 dataset is part of the training data of the BEA-2019 shared task. Apart from English sentences, the Lang-8 dataset also released corrections in other langua

W&I+LOCNESS: The Write & Improve (W&I) and LOCNESS corpus [9] contains 3600 essays (∼755k) from international language learners (A1-C2) and 100 essays (∼46k) from native British/American English speakers. The data comes from the Write & Improve online platform [3] [80] and the LOCNESS corpus [24], with 55 error types extracted automatically. The dataset is divided into a training set that only contains sentences from language learners and development and test sets that combine L1 and L2 speakers. The W&I+LOCNESS corpus is one of the training sets in the BEA-2019 shared task, while its development and test sets are the official ones for the task.

Non-English Datasets

Arabic: The Qatar Arabic Language Bank (QALB) [86] is the main effort to collect annotated corpora for Arabic GEC. Data coming from the project was used in two shared tasks: QALB-2014 [43] and QALB-2015 [85]. The former corresponds to 21.3k documents (∼1.1M words) obtained from comments on a news website, being split into train, development and test sets. The latter corresponded to 920 documents (∼48.5k words) from news comments and 633 essays (∼90.5k words) from L2 learners (A1-C2). Different from other GEC datasets that tend to work at the sentence level, the QALB corpus is provided at the document level. The edits are extracted by trained annotators, and classified into 7 categories automatically.

Chinese: For Chinese we bring attention to the NLPCC-2018 [88], the first shared task on full error correction in Mandarin Chinese. The training data comes from Lang-8 user submissions and comes to a total of 717k sentences (∼14.1M characters). As was mentioned for English data from Lang-8, the annotators are not professionals and the corrections tend to be noisy. For the test split, 2000 sentences (∼61.3k characters) written by international university students from the PKU Chinese Learner Corpus as used. The errors were manually classified into 4 error types.

Czech: The AKCES-GEC corpus [49] has 47.3k sentences (∼505k words) by L2 learners and Romani children who speak a Czech ethnolect as their first language. The data

[3] https://writeandimprove.com/

comes from the Czech Language Acquisition Corpora (AKCES) which is comprised of essays and exam scripts. The sentences are manually annotated by specialists and classified into 25 error types. The dataset has train, development and validation splits.

Follow-up work to AKCES-GEC, the Grammar Error Correction Corpus for Czech (GECCC) [45] extends the dataset by including formal texts by L1 students, informal discussions from social media and news articles. The corpus has a total of 83k sentences ($\sim$949k words) that were manually annotated. Notice that the original data was re-annotated to keep a consistent style. An automatic approach was used to classify the errors into 65 categories. This is one the largest non-English datasets with manually annotated sentences, being itself larger than most English datasets. Similarly to AKCES-GEC, GECCC is also provided with official training, development and testing splits

German: The Falko-MERLIN corpus [5] consists of 24k sentences ($\sim$381k words) written by language learners from a wide range of levels (A1-C2). The two data sources for the dataset are the Falko [59] and MERLIN [6] corpora, which consists of essays and exam scripts, respectively. For both corpora, sentences were manually annotated by specialists and 56 error types were extracted automatically. The dataset is split into train, test and validation.

Russian: The Russian Learner Corpus of Academic Writing (RULEC) [36] is comprised of 12.5k essays ($\sim$206k words) written by L2 university students and Russian speakers in the United States of America. The dataset was manually annotated with 23 error types and is divided into train, development and test.

Turkish: Recent work on developing automatic annotation techniques in Turkish [70] collected a dataset with 106 sentences ($\sim$800 words). The data comes from academic studies on errors made by language learners when writing in Turkish. Due to its very small size, the dataset is better suited for testing only.

2.5.2 Evaluation Metrics

In order to evaluate the performance of a model in a given annotated corpus the research community relies on automatic evaluation metrics. Even though some nuances are lost when not performing human evaluation, the automated metrics are able to reflect the performance and capture if the correction follows the general guidelines of the corpora. Now we will introduce the two main metrics used in GEC, Max Match (M^2) and ERRANT.

Max Match (M^2)

Max Match [18] is a reference-based metric, that is, it considers the original sentence, a reference correction and a hypothesis sentence predicted by the model. It then extracts the edits and counts them as True Positive (TP) if the edit is in both reference and hypothesis, False Positive (FP) if the edit is only in the hypothesis, or False Negative (FN) if the edit is in the reference but not in the hypothesis. Using these counts Precision (P), Recall (R) and F_β-score are computed as follows

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \qquad \mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad \mathrm{F}_\beta = (1 + \beta^2) \cdot \frac{\mathrm{P} \times \mathrm{R}}{\beta^2 \cdot \mathrm{P} + \mathrm{R}}$$

The usual choice is $\beta = 0.5$, which weighs precision twice as much as recall. This reflects the fact that in GEC we are more interested in being precise than in correcting all the errors in the sentence. When multiple references are provided the best scores for each sentence is chosen to compute the overall metric.

The main challenge when computing the TP, FP, and FN is edit overlap. For example, the edit [*love plays* $\rightarrow$ *loves playing*] could also be expressed as the two edits [*love* $\rightarrow$ *loves*] and [*plays* $\rightarrow$ *playing*]. Because of that, if we naively compare reference and hypothesis edits it may result in an incorrect attribution of FP and FN. To solve this issue, M^2 computes the Levenshtein alignment between the original sentence and the predicted hypothesis to find the edits that maximize the match to the reference edits.

ERRANT

The main problem with the M^2 is that the results do not provide a way to know which types of errors the models are good/bad at correcting. To tackle this problem ERRANT [10] computes precision, recall and F-score individually for each type of error. The errors are categorized in two separate ways:

- Operation (3 labels): Missing, Replacement, and Unnecessary
- Error type (25 labels): Adjective, Orthography, Verb Form, Verb Tense, etc.

Thus, for example, ERRANT allows us to evaluate only replacement errors (R), only adjective errors (ADJ), or replacement adjective errors (R:ADJ). Such categories are automatically generated for the edits given a paired dataset. In practice, ERRANT extracts

Original	They	love	to	playing	footbal	in	backyard	.
Binary	C	C	I	C	I	C	I	C
Operation	C	C	U	C	R	C	M	C
Type	C	C	PREP	C	SPELL	C	DET	C
Operation-Type	C	C	U:PREP	C	R:SPELL	C	M:DET	C

Table 2.3: Example of different levels of granularity in GED.

edits using a linguistically-enhanced version of the Damerau-Levenshtein algorithm and classifies them into different types using a rule-based system. Such rules are hand-crafted using linguistic knowledge of the language of interest. Although the original version of ERRANT only supported English new versions have been proposed to support German, Greek, Arabic, and Czech.

2.6 Grammatical Error Detection

Grammatical Error Detection (GED) is the task interested in detecting the location of errors in a sentence. It can be seen as a classification counterpart of the GEC task, or even as a step that correction systems have to undergo. Similarly to the case in GEC, the definition of grammatical error in GED is open and subjective to corpus-based conventions.

Formally, GED is usually formulated as a word-level classification task. That is, for every word in the input sentence we are interested in indicating which are the words that make the sentence incorrect. Importantly, in the case of a missing word, the incorrect label is attributed to the word to its right. Classification in GED can be performed in binary or multiclass fashion, and the latter can be performed to different levels of granularity: operation, edit and operation-edit (An example is provided in Table 2.3).

Evaluation of GED is performed using Precision, Recall and $F_{0.5}$-score. When the multiclass approach is performed the metrics are computed using macro averages of the classes.

It is a mistake to believe that there is no need to research GED because it can be seen as a simpler version of GEC. First of all, some applications are already more well-suited for GED, for example, quality assurance in proofreading, where knowing about the existence of errors (and potentially their type) is enough. Secondly, there are multiple ways in which GED can be explicitly used to improve GEC, which means that improvements to detection can directly impact advances in correction. Finally, because in general classification a simpler than generation, studying GED can serve as an efficient way to explore ideas that can also be applied to GEC.

2.6.1 Datasets

Language	**Source Corpus**	**Nr. sentences**	**Nr. tokens**	**Nr. erros**	**Error rate**
Czech	GECCC	35.4k	399.7k	84k	0.210
English	FCE	33.2k	531.4k	50.8k	0.096
English	REALEC	8.1k	177.8k	16.6k	0.093
German	Falko-MERLIN	24k	381.1k	57.9k	0.152
Italian	MERLIN	7.9k	99.7k	14.9k	0.149
Swedish	SweLL-gold	8.5k	145.5k	27.3k	0.187

Table 2.4: Statitics of the datasets from MultiGED-2023. Notice that English-REALEC only has dev and test splits.

Conceptually, building a dataset for GED does not require having a corrected version of the sentence. An annotator could simply label each word according to the desired granularity level. However, in practice, GED datasets are commonly extracted trivially from already-existing GEC corpora. For this reason, there is a large overlap between GED and GEC datasets.

The vast majority of methods developed for GED focused only on English. In those works training and evaluation were always performed using the FCE dataset converted for the detection task. More specifically, using the error span provided by the GEC annotations we can mark exactly which words are incorrect or tag it with more specific error types. The main focus is on binary GED, but some methods also experiment with finer granularity. Some other GEC corpora, such as NUCLE, have also been explored as additional training data.

Focusing on expanding GED beyond English, the MultiGED-2023 shared task [72] released a collection of multilingual datasets for GED. Apart from English it also covers Czech, German, Italian and Swedish. For every language, the shared task has a dataset that is used for training, development and testing. The only exception is English, for which a second dataset is provided for development and testing. The task standardized all datasets as word-level binary classification providing the basis for a unified study of GED systems in different languages.

For English, the MultiGED task uses FCE for training and evaluation and RULEC only for evaluation. The latter was introduced with the shared task and is composed of essays written by L1 Russian speakers with B1-B2 level. For German and Czech a processed version of the GEC datasets Falko-MERLIN and GECCC were used, respectively. For Italian, train, development and validation sets were created using data from the MERLIN corpus. Similarly, MultiGED processed Swedish data from the Swell-gold corpus [73] to make it suitable for the shared task. A summary of all the datasets is provided in Table 2.4

Chapter 3

Related Work

3.1 Grammatical Error Detection

Similar to other areas of NLP, early work on GED has been performed through classical statistical methods [13] [23]. With the breakthrough in deep learning, LSTM started being used to train detection systems [56]. Various approaches using RNNs have been developed, exploring auxiliary loss functions [57] [55], learning to generate synthetic errors [34], and using representations from pre-trained models as additional information [4]. Naturally, with the increase in popularity of the Transformer architecture, it began to be used for GED. [32] introduced an architecture in which the classification head is connected to all layers of the Transformer to allow the model to focus on lower-level features. Later [84] showed that fine-tuning a pre-trained ELECTRA exhibits better performance as its pre-training objective is similar to GED.

On the multilingual side, the MultiGED-2023 shared task [72] was the first effort to unify datasets in different languages for GED. There were two main submissions to shared tasks that are relevant to our work. EliCoDe [15] was the winner in five out of six test sets in the task, only losing in the out-of-domain English dataset (REALEC). Their best models are XLM-RoBERTa Large trained in each of the languages. The work compared the performance of mono and multilingual models, but only used a single random seed and the models were trained using a batch size of 4. The second submission was DSL-MIM-HUS [37], which used a model trained with multilingual data and achieved the best result for the REALEC test set. However, it is important to point out that the model used the validation data for REALEC as part of the training set. That is not against the rules of the shared task but defeats the purpose of evaluating cross-dataset generalization.

3.2 Grammatical Error Correction

3.2.1 Supervised Methods

Supervised GEC can be broken into two types of methods. The first is treating correction as neural machine translation (NMT), where we are interested in "translating" from incorrect language to correct language. Within this framework, GEC is treated as a sequence-to-sequence task. The other approach is thought tagging, where each word in the input sentence is tagged with an edit operation that should be applied to generate the correction. These methods treat GEC as a token-classification problem. Let us now cover the relevant work in both areas.

Correction as Neural Machine Translation

After the success of RNNs with attention to machine translation, [83] proposed their usage for GEC. In addition, RNN methods that operated on character-level [77] and with both words and characters simultaneously were also developed [30].

Later, the first NMT method for GEC using the Transformer was introduced [31]. Variations with copy mechanism were also explored [88][29], as in correction most of the output comes directly from the input. [33] was the first method to use representations extracted with BERT to perform Transformer-based GEC. Pre-trained sequence-to-sequence Transformers, such as BART [35] and T5 [60] were also used for correction in English, German, Czech and Russian.

As GEC datasets are relatively small, efforts were made to obtain more data efficiently. Such datasets are used to start training and later the model is fine-tuned with higher quality data. One option is to use filtered Wikipedia edits to obtain more data [25] [5]. Alternatively, [27] proposed the use of a spell checker to introduce errors in news article sentences, which [49] expanded to German, Czech and Russian. To avoid determining language-specific parameters, gT5 [60] uses general character-level and token-level corruptions to create a GEC version of the mC4 dataset [79] for multilingual pre-training.

Detection can be thought as an implicit part of correction, but some methods make explicit use of GED as part of the GEC pipeline. [84] included a second encoder to the Transformer that receives as input the GED tags predicted by a detection model. TemplateGEC [39] uses GED tags to construct correction templates that resemble the pre-training strategy of T5.

Correction as Edit Tagging

The tagging approach started with LaserTagger [42], which used a BERT encoder to predict edits operations (*keep*, *remove*, and *add*) and a single-layer decoder that applies the edits. Parallel Iterative Edit (PIE) [3] proposed a encoder-only method that predicts a larger set of edit operations that are automatically applied. To allow multiple corrections in the same location the sentence is passed through the model iteratively until no error is found.

GECToR [50], another encoder-only model, improved generalization by introducing the concept of g-transformation tags, which perform task-specific operations (case change, merge words, verb forms, noun count, etc). GECToR also trains a GED head that is used during inference to decide if a correction should be applied. The g-transformations are language-specific, to solve this problem [65] proposed a method for automatically extracting transformations from paired data, which was used in German, Czech and Russian.

3.2.2 Unsupervised Methods

The study of unsupervised GEC studies how to perform correction without using models trained on paired datasets. This is important because collecting datasets is an expensive process, and various languages have large amounts of raw text available.

The first strategy relies on the assumption that an incorrect sentence probability is lower than its corrected version. [8] used a 5-gram language model to assess sentence probability and performed greedy search over candidates generated using a spell checker and an inflectional morphology database. MAGEC [26] evaluated the same strategy but using a Transformer language model and included experiments in German and Russian. The approach was also investigated by [1] using pre-trained language models such as BERT, GPT, and GPT-2.

Alternatively, a LLM can be used with a prompt to perform GEC. [76] showed that ChatGPT's performance is worse than supervised methods, however using a sample of only 300 examples. [22] arrive at the same conclusions exploring GPT-3.5 on complete datasets in English, German, and Chinese. Interestingly, [17] shows that both GPT3.5 and GPT-4 achieve better result than supervised models when performing human evaluation studies.

Chapter 4

Multilingual Grammatical Error Detection

Grammatical Error Detection's goal is to identify which words are wrong in a sentence. It is an important application for language learning assistive tools and can also be used to improve Grammatical Error Correction systems. Given such benefits, exploring and improving the performance of GED models in a wide variety of languages is of great importance. To this end we study different approaches to multilingual GED, being the first to evaluate a set of capabilities of these models.

The MultiGED-2023 shared task was the first effort toward multilingual GED, having six datasets in five different languages. The task was open in terms of models allowed, encouraging the development of multilingual models but still allowing monolingual solutions to be submitted. As a result of that, the top entry in the competition was a collection of models trained on monolingual data, surpassing the multilingual model approach.

More specifically, we refer to models trained with a single language dataset as monolingual but it is important to notice that the pre-trained architecture itself was a multilingual XLM-RoBERTa. This happens because there are no monolingual pre-trained models for some of the languages in MultiGED. In contrast, the models that we refer to in this work as multilingual are trained on a multilingual collection of GED datasets. Figure 4.1 illustrates the difference between the two approaches.

The top entry in the MultiGED task, EliCoDe, used one monolingual model per language. In their paper, they explored several approaches to train the XLM-RoBERTa Large model used. In addition to the monolingual model they also investigated a multilingual solution, but it was not used in the competition as it showed worse overall results in their

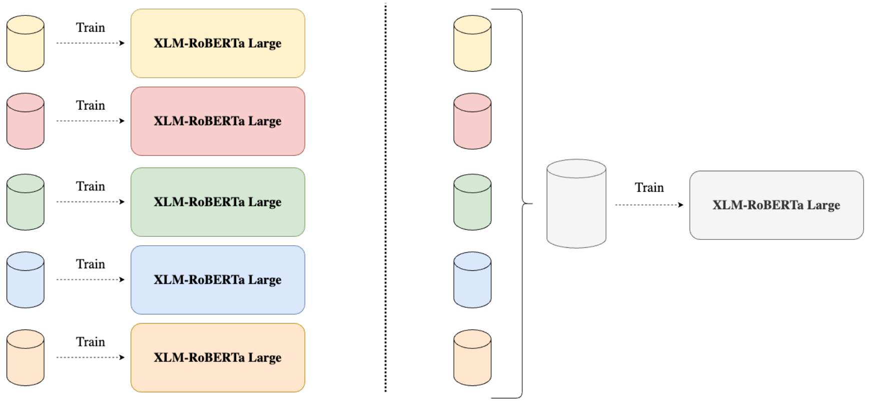

Figure 4.1: Difference between a collection of models trained on monolingual data (left) and a single model trained with multilingual data (right).

experiments. However, the difference between the monolingual and multilingual models was very small and could have been in favour of any model when varying the random factors involved in training. This is amplified by the fact that the models were trained using very small batch sizes and the datasets are relatively small, increasing randomness.

Another important remark regarding the MultiGED shared task is regarding the REALEC dataset, an English dataset comprised of sentences written by native Russian speakers. This dataset has only validation and test splits, therefore the performance relied completely on the generalization from the other training sets. For this reason the results were considerably lower than the other datasets. It raises more questions about how powerful the GED models are in terms of generalizing to new datasets in the languages already seem during training.

One aspect that is not covered by the MultiGED shared task is cross-lingual generalization, more specifically in a zero-shot fashion. This corresponds to testing a model in new languages that were not seen during training. As long as the models, including the ones trained on monolingual data, are pre-trained and have embeddings for multiple languages it is possible and reasonable to evaluate zero-shot cross-lingual capabilities. Having good generalization in this sense allows applying GED techniques to languages that do not have available training sets.

Our goal is to further investigate the difference between models trained with monolingual and multilingual data. To this end, we conduct experiments along the same lines as EliCoDe but following a more rigorous experimental setting. In addition, we explore the performance of both approaches in terms of generalization to new datasets, both in the same language as the training set and for new languages in a zero-shot setting.

4.1 Methodology

We are interested in studying the difference between monolingual and multilingual models through three different perspectives: in-domain, cross-dataset, and cross-lingual generalization. To this end, we extended the MultiGED task by creating different GED datasets using existing GEC corpora. Importantly, we focus on conclusions that take into consideration the intrinsic randomness in training the models. Thus, we performed all the experiments using multiple random seeds and more reasonable hyperparameters.

First, we evaluate the performance of the monolingual and multilingual models in a hold-out split of the dataset in which they were trained. The idea is to replicate the experiments performed by EliCoDe but with a more robust setup. For this experiment, the main interest is to find out if there is any difference between the performance of monolingual and multilingual models when the test data follows the same distribution seen during training.

Secondly, we study how the same models perform when the test data is on a language presented during training but comes from different datasets. This is relevant because different datasets have different error distributions, not only in terms of frequency but also the types of errors. Motivated by the results of the REALEC dataset in the MultiGED competitions, we are interested in observing if multilingual models show better generalization than their monolingual counterparts. We use the same models trained on the MultiGED training sets. Unfortunately, there are no extra datasets available for all languages that could be used for GED. Thus, this study is only performed in English and Czech.

Finally, we are interested in observing how well the GED models generalize for new languages never seen during training. For this experiment, we processed datasets in four new languages: Arabic, Chinese, Russian and Turkish. The set of languages chosen is based on two things: data availability and language diversity. The former is necessary because only a limited set of languages have a corpus with GED or GEC annotation. The latter is relevant to see the impact of syntax and morphology diversity when transferring to new languages. Hence, we selected languages from four different language families, three of them not present in the MultiGED training set.

Language	**Language Family**	**Source Corpus**	**Nr. sentences**	**Nr. tokens**	**Nr. erros**	**Error rate**
Czech	Indo-European	AKCES-GEC	2.5k	28k	6.4k	0.228
English	Indo-European	W&I+LOCNESS	4.4k	87k	8.4k	0.096
Arabic	Afro-Asiatic	QALB-2014	1k	53.9k	16.7k	0.291
Chinese	Sino-Tibetan	NLPCC-2018	2k	40.2k	4.8k	0.120
Russian	Indo-European	RULEC-GEC	2.5k	41.2k	2.5k	0.060
Turkish	Turkic	Uz & Eryiğit	105	822	195	0.237

Table 4.1: Statistic of the new GED datasets used for evaluation. Note that the Arabic dataset is split into documents rather than sentences.

4.2 Experimental Setup

Using the MultiGED training sets we train one monolingual model in each language and a single multilingual model trained on the union of all training sets. We refer to the monolingual models by the language of their training data. For all models, we use the pre-trained XLM-Roberta Large model from HuggingFace, fine-tuning all weights and a binary classification head. The networks are optimized using AdamW, learning rate 1e-5 with linear scheduling, and batch size of 32. The training is interrupted when the validation performance stops increasing for 3 epochs.

We made use of existing GEC corpora to create new GED datasets for evaluation. For cross-dataset generalization W&I+LOCNESS and AKCES-GEC datasets were used for English and for Czech respectively. For cross-lingual evaluation, we used QALB-2014 for Arabic, NLPCC-2018 for Chinese, RULEC for Russian, and Uz & Eryiğit for Turkish. The statistics of all the datasets in the context of binary GED are presented in Table 4.1.

All the results indicated in this section come from the development sets of the datasets unless stated otherwise. Naturally, evaluation was performed using Precision, Recall and $F_{0.5}$-score. We report the average across all five random seeds and the corresponding standard deviation for all experiments.

4.3 Results and Analysis

4.3.1 In-domain

For the in-domain test sets, the results from Table 4.2 demonstrate that there is no difference in overall performance between the monolingual models and the multilingual one.

	Monolingual			Multilingual		
Language	Prec	Rec	$F_{0.5}$	Prec	Rec	$F_{0.5}$
English	73.89 ± 0.81	52.71 ± 1.23	68.38 ± 0.33	74.70 ± 1.18	52.03 ± 1.83	68.69 ± 0.25
Czech	83.79 ± 0.36	56.73 ± 0.57	76.49 ± 0.15	84.51 ± 0.35	55.32 ± 0.95	76.44 ± 0.24
German	85.40 ± 0.40	71.67 ± 0.97	82.24 ± 0.15	85.44 ± 0.47	72.05 ± 0.69	82.38 ± 0.28
Swedish	84.03 ± 0.44	60.82 ± 1.19	78.07 ± 0.27	83.58 ± 0.74	61.48 ± 1.34	77.97 ± 0.44
Italian	85.84 ± 0.94	62.76 ± 2.04	79.94 ± 0.34	85.51 ± 0.56	64.93 ± 1.63	80.40 ± 0.38

Table 4.2: Results of monolingual and multilingual models on the in-domain datasets.

The average $F_{0.5}$-score is higher for the multilingual model in three out of five languages (English, German and Swedish). However, the difference between the models is small and falls within the standard deviation of the scores. Importantly, the same behaviour is exhibited by Precision, Recall, and, $F_{0.5}$-score. For those reasons, we can affirm that for the in-domain evaluation, multilingual and monolingual models achieve the same performance. This is interesting because it shows that the models are able to learn multiple sets of grammatical rules and automatically switch between them when a sentence is passed through the model, even though the language of the input sentence is not given explicitly.

4.3.2 Cross-dataset

For these experiments, we are interested in observing the performance of GED models when evaluated in languages seen during training but with data coming from a different dataset. Observing the results from Table 4.3 it is possible to notice that the results are different for different datasets. For the REALEC (English) and AKCES-GEC (Czech) datasets we observe that the same phenomena observe the same as in the in-domain evaluation. The multilingual model presents better results on average, but the difference is small and falls within the standard deviation from both models. On the other hand, for W&I+LOCNESS (English) we can see a clear advantage of the multilingual model. For this dataset, we can see that the higher $F_{0.5}$-score is mainly because of the higher precision achieved by the model.

4.3.3 Cross-Lingual

Here we compare how well models perform in languages that were not seen during training. Recall that even the models trained on monolingual models use a pre-trained multilingual

Dataset	Monolingual Prec	Monolingual Rec	Monolingual $F_{0.5}$	Multilingual Prec	Multilingual Rec	Multilingual $F_{0.5}$
REALEC	46.48 ± 0.59	40.36 ± 0.59	45.11 ± 0.32	47.17 ± 1.23	40.38 ± 1.29	45.61 ± 0.59
W&I+LOCNESS	65.61 ± 0.72	39.02 ± 0.79	57.73 ± 0.48	68.62 ± 1.49	38.88 ± 1.16	59.49 ± 0.46
AKCES-GEC	87.21 ± 0.17	61.88 ± 0.36	80.61 ± 0.12	87.95 ± 1.03	61.25 ± 1.04	80.88 ± 0.45

Table 4.3: Results of monolingual and multilingual models on the cross-dataset evaluation.

architecture, therefore they can be tested in different languages. As we can observe in Figure 4.2, the model trained on multilingual data shows better generalization to novel languages. The only language with different behaviour is Chinese, where all models seem to struggle and exhibit poor performance. Overall, all models tend to have higher variance results when compared to in-domain and cross-dataset evaluations.

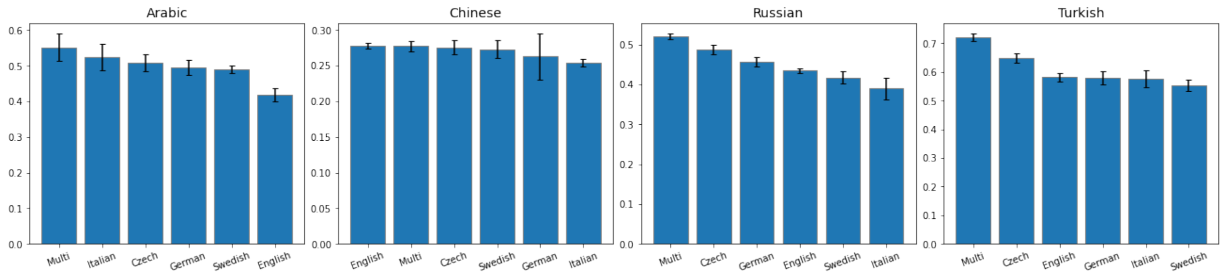

Figure 4.2: Results of cross-lingual transfer across all models.

A relevant comment can be made regarding the influence of dataset size. For Arabic, the best monolingual model is the one trained in Italian while the worst is the English model. Interestingly those are respectively the smallest and second largest corpora in our training sets. On the other hand, for Russians, the Italian model demonstrates the worst performance, which indicates that other factors such as error distribution and language similarity may also be a factor.

Chapter 5

Detection Assisted Prompt-Based Grammatical Error Correction

With the recent advances in Large Language Models in zero-shot tasks the various fields of NLP have put effort in making use of this technology. The main idea is to make use of the incredible generalization and flexibility that arise from training billions of parameters on trillions of tokens. By shifting the paradigm from finetuning with annotated datasets to prompting an LLM with task instructions, the power of AI applications has become accessible to a large number of people. Naturally, a relevant area such as Grammatical Error Correction has begun to be explored in the context of LLMs.

Language models have already been explored for unsupervised GEC over the past decade. The main focus of this line of work was on using the probability of a sentence before and after edits to assess correctness. However, it is important to establish the distinction between those works and the use of LLM thought prompting. When referring to LLM-GEC we are interested in performing GEC by prompting a LLM, most often through the use of natural language instructions. In addition, in-context examples can be provided to help the model perform the task.

Previous work has investigated the capabilities of the GPT-3.5 and GPT-4 in GEC in different languages. In general, the results tend to indicate that such systems have reasonable performance but are still far from supervised approaches when evaluated with automated metrics. This comes from the fact that the model tends to exhibit large values for False Positives, which is often a product of either overcorrection or alternative solutions.

A model overcorrects when it changes the sentence in a region that was not wrong to start with. It can be due to addition, replacement or deletion of words. Notice also, that overcorrections may change the meaning of the sentence, which goes against the principle of GEC. But we also consider it to be overcorrection even if the meaning is preserved, as GEC is usually defined in terms of minimal edits to make the sentence correct.

On the other side, an alternative solution happens when a correction is right but is not the one expected by the annotated dataset. This is a direct consequence of GEC being an ill-posed problem and is amplified because there is no training for the model to learn the style of the corpus. One may argue that this is only a problem when we rely on automatic metrics, as they only know the truth in the dataset. In fact, LLM tends to be competitive with supervised systems under human evaluation. However, in various applications, we are interested in controlling the universe of possible corrections. It can be viewed as following a style guide that is determined by the annotated corpus.

Simple solutions to these problems include tweaking the instruction and increasing the number of in-context examples provided in the prompt, but it has not been shown to reduce or improve the results by a large margin. We hypothesize that by making use of a GED system that can indicate if corrections need to be made the problem of overcorrection can be mitigated. Similarly, we are interested in understanding if a better selection of in-context examples can improve the model's ability to follow the style of the dataset.

Concretely, we studied how GED models studied in Chapter 4 can be used to assist LLM-based GEC and mitigate the overcorrection and alternative solutions problems. Two different ways to improve correction with detection are studied: filtering and in-context example selection. Similarly to [49] [60] [65] we investigate datasets in English, Czech, German and Russian and compared them with supervised baselines. As part of the experiments we are, to the best of our knowledge, the first to benchmark Llama2 on English GEC and to evaluate GPT3.5 in Czech and Russian.

5.1 Methodology

5.1.1 GED Filtering of Correct Senteces

In order to mitigate the overcorrection problem we propose using a multi-stage pipeline in which sentences are first passed through a GED model. The result of the detection module determines if the sentence needs to be passed through the LLM for correction. If a sentence is classified as correct by the GED model, it is simply copied as output, without being fed to the LLM. This takes advantage of the fact that GED models can be trained with way less data than supervised GEC systems.

Dataset	Language	Prec	Rec	$F_{0.5}$
FCE	English	92.84 ± 0.33	78.94 ± 1.91	89.67 ± 0.25
W&I+LOCNESS	English	89.85 ± 0.85	69.78 ± 2.11	84.94 ± 0.08
Falko-MERLIN	German	96.40 ± 0.27	91.68 ± 0.50	95.42 ± 0.18
AKCES-GEC	Czech	97.00 ± 0.30	86.66 ± 01.00	94.73 ± 0.24
GECCC	Czech	95.11 ± 0.16	85.69 ± 0.75	93.06 ± 0.19
RULEC	Russian	75.70 ± 1.33	64.48 ± 1.02	73.14 ± 0.88

Table 5.1: Results of the multilingual GED model when evaluated on a sentence level. For this experiment, the sentence is considered incorrect if any of its words are classified as incorrect.

Notice that this framework only requires sentence-level detection, as we are interested in filtering entire sentences out. Thus, one could train GED models that give a single label to the whole sentence. An alternative approach is to perform word-level detection and aggregate the results producing a sentence classification. As we can see in Table 5.1 the multilingual model explored in Chapter 4 exhibits great sentence-level performance in our languages of interest.

5.1.2 GED for In-context Example Selection

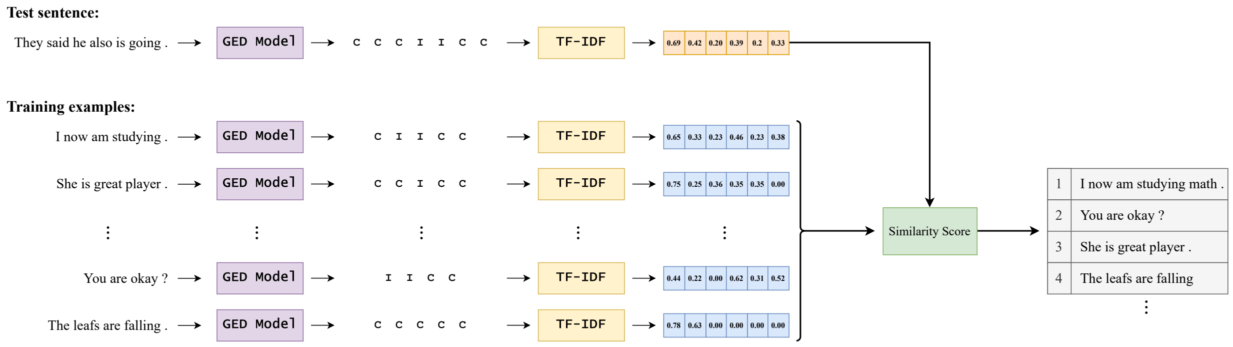

Figure 5.1: Illustration of the TF-IDF GED framework proposed for in-context example selection. First, the sentences are passed to a GED model to extract the detection tags, and then TF-IDF is used on the tags to extract feature vectors. Once with the feature vectors, we compute the similarity between the test sentence and the training examples to obtain the in-context examples.

Previously work in NLP tasks using LLM demonstrated that the selection of in-context examples can have a big impact on performance [40] [87] [66]. By having more informative examples in the prompt, the model can better capture the essence of the task. Thus, investigating these methods can improve LLM-based GEC by generating corrections that are better at following the dataset style.

Different approaches use semantical similarity between the input and examples in the training set to select the ones that are most helpful to perform a given task [40] [66]. However, in GEC we are more interested in structural similarities than higher-level features such as semantics. Two sentences can have high semantical similarity but provide no help to correct each other. To solve this problem we propose using information obtained from a word-level GED system to compute similarities.

Given two input sentences $\boldsymbol{x^i} = (x^i_1, ..., x^i_N)$ and $\boldsymbol{x^j} = (x^j_1, ..., x^j_M)$ we extract their detection labels using a GED model obtaining $\boldsymbol{l^i} = (l^i_1, ..., l^i_N)$ and $\boldsymbol{l^j} = (l^j_1, ..., l^j_M)$. Using the label sequences, we use term frequency-inverse document frequency (TF-IDF) to compute feature vectors. Notice that the terms in this case are n-grams of predicted detection labels, not the words in the sentences. After applying TF-IDF to extract fixed-sized feature vectors we compute the cosine similarity between the pairs. The framework which we refer to as TF-IDF GED is illustrated in Figure 5.1.

In principle, there is no constraint to the granularity of the detection labels. Current GED systems are better at binary detection but finer classification of the error types should improve the example selection pipeline. Another relevant hyperparameter of the method is the n-gram range of the TF-IDF, which controls how local are the error structures used to represent the sentences.

5.2 Experimental Setup

We are interested in studying GED filtering and TF-IDF GED in English, Czech, German and Russian using powerful LLMs. For English, we perform the experiments using Llama2. For the other languages, we use GPT3.5, more specifically `gpt-3.5-turbo`, as it has been used for non-English GEC with reasonable results. To investigate the effectiveness of our methods, we first run baselines for all languages. We run the models using 0, 1, 2, 4 and 8 in-context examples in Enlgish and 0 and 8 for non-English datasets. The prompt used in our experiment comes from [17] with minimal alterations to facilitate parsing the answers:

For the non-English experiments, we performed some early experiments and there was no difference between prompting the model in the language of the sentence or simply

```
 Reply with a corrected version of the input sentence delimited by <input> </input>
with all grammatical and spelling errors fixed. If there are no errors, reply with
a copy of the original sentence. Output the corrected version of the sentence
delimited by <output> </output> tags directly without any explanations. Please
start:
<input>She like to walks her dog</input>
<output>
```

specifying the language in the English prompt (*"Reply with a corrected version of the German input sentence..."*, for example). Therefore, we chose to run the experiments using the English prompts with specified languages.

Following the same approach as previous work [49] [60] [65], we used W&I+LOCNESS for English, Falko-MERLIN for German, AKCES-GEC for Czech, and RULEC for Russian. The training splits of all the datasets are used as in-context examples for prompting. For the experiments, we use the development sets when exploring the different methods and hyperparameters, and the test sets are used on the best model found on validation.

The core component of the methods is the GED model. Given all the results obtained in Chapter 4 we use an XLM-RoBERTa Large trained with the multilingual dataset. Importantly, the multilingual GED model was trained on the same German dataset, on different English and Czech corpora, and was not trained on Russian. The model was evaluated on all datasets and exhibited good results in both sentence-level and work-level binary classification.

In the experiments for the in-context example selection, we compare our proposed TF-IDF GED against the semantical similarity approach. For our method, we use TF-IDF with n-grams from 1 to 4 elements. For semantical similarity we use Sentence-BERT (SBERT) [58], more specifically the `all-MiniLM-L6-v2` model from HuggingFace [1], to extract feature vectors.

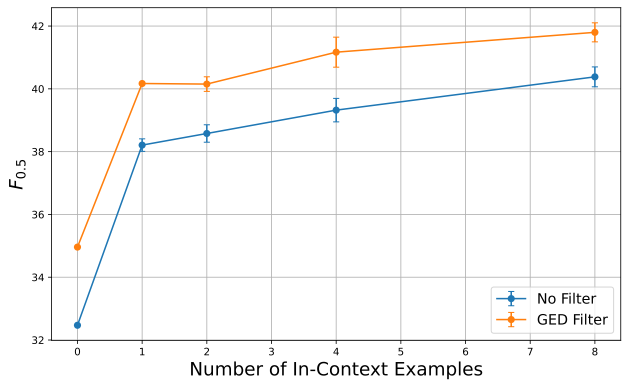

Figure 5.2: Effect of filtering out correct sentences on the W&I+LOCNESS development set.

5.3 Results and Analysis

5.3.1 GED Filter

Compared to directly passing all sentences to the LLM, filtering out correct sentences shows a positive effect on the results in English. As it is shown in Figure 5.2 the gains are noticed regardless of the number of in-context examples. Table 5.2 show all possible variations of the model, including the TF-IDF GED that will be discussed in the next subsection. Observing the results in Table 5.2 that correspond to performing GED Filter, we can see that the strategy increases the precision of the model, resulting in a better $F_{0.5}$-score.

As we can see in Table 5.3 filtering also improves the results in the non-English datasets. Interestingly, the largest gain was in Russian, a language that is not even present in the training data of the GED model used for filtering. This is good because we are able to improve unsupervised correction using only cross-lingual capabilities of the GED model.

Notice that by filtering out sentences predicted as correct we not only improve the score of the model, but we also reduce the number of times that the LLM is prompted. In a practical setting, where API calls to the LLM are often charged on a token basis, having a lightweight GED perform filtering can increase both the performance and cost efficiency of the pipeline. For this reason, it would be interesting to study ways to reduce the size of the GED models, using distillation and quantization techniques.

[1]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

#-Shots	GED Filter	TF-IDF GED	Prec	Rec	$F_{0.5}$
0			31.22 ± 0.11	$\mathbf{38.71 \pm 0.26}$	32.47 ± 0.14
	✓		$\mathbf{35.53 \pm 0.12}$	32.86 ± 0.16	$\mathbf{34.96 \pm 0.10}$
1			38.68 ± 0.27	$\mathbf{36.42 \pm 0.12}$	38.21 ± 0.20
	✓		42.78 ± 0.31	31.29 ± 0.08	39.85 ± 0.21
		✓	40.94 ± 0.17	34.51 ± 0.09	39.47 ± 0.14
	✓	✓	$\mathbf{44.74 \pm 0.23}$	30.14 ± 0.08	$\mathbf{40.79 \pm 0.17}$
2			39.66 ± 0.28	$\mathbf{34.78 \pm 0.32}$	38.58 ± 0.28
	✓		43.90 ± 0.28	29.93 ± 0.23	40.15 ± 0.23
		✓	43.85 ± 0.15	33.69 ± 0.19	41.35 ± 0.16
	✓	✓	$\mathbf{46.24 \pm 0.15}$	30.40 ± 0.22	$\mathbf{41.88 \pm 0.18}$
4			40.54 ± 0.40	$\mathbf{35.11 \pm 0.30}$	39.32 ± 0.37
	✓		45.25 ± 0.59	30.24 ± 0.26	41.17 ± 0.48
		✓	45.46 ± 0.17	32.19 ± 0.21	42.00 ± 0.14
	✓	✓	$\mathbf{47.54 \pm 0.07}$	29.88 ± 0.19	$\mathbf{42.51 \pm 0.11}$
8			41.81 ± 0.35	$\mathbf{35.52 \pm 0.34}$	40.38 ± 0.32
	✓		46.02 ± 0.41	30.58 ± 0.23	41.80 ± 0.30
		✓	47.67 ± 0.24	31.82 ± 0.23	$\mathbf{43.35 \pm 0.24}$
	✓	✓	$\mathbf{49.44 \pm 0.25}$	30.16 ± 0.23	$\mathbf{43.83 \pm 0.25}$

Table 5.2: Complete results on the W&I+LOCNESS development set. Checkmarks (✓) indicated if the corresponding method was used. The reported scores are the average between five random seeds, with indicated standard deviation. Bold values indicate the highest score for each number of in-context examples.

Language	#-Shots	GED Filter	TF-IDF GED	Prec	Rec	$F_{0.5}$
Czech	0			72.33 ± 0.14	$\mathbf{64.56 \pm 0.15}$	70.63 ± 0.13
		✓		$\mathbf{75.55 \pm 0.10}$	61.31 ± 0.15	$\mathbf{72.20 \pm 0.11}$
	8			76.73 ± 0.18	$\mathbf{65.18 \pm 0.10}$	74.10 ± 0.13
		✓		$\mathbf{79.08 \pm 0.17}$	62.23 ± 0.12	75.02 ± 0.13
			✓	78.01 ± 0.34	64.65 ± 0.21	74.91 ± 0.24
		✓	✓	$\mathbf{79.46 \pm 0.15}$	62.50 ± 0.16	$\mathbf{75.37 \pm 0.11}$
German	0			65.47 ± 0.18	$\mathbf{63.73 \pm 0.21}$	65.11 ± 0.14
		✓		$\mathbf{68.53 \pm 0.20}$	60.89 ± 0.10	$\mathbf{66.86 \pm 0.13}$
	8			69.58 ± 0.12	$\mathbf{63.73 \pm 0.23}$	65.11 ± 0.12
		✓		71.63 ± 0.20	59.56 ± 0.18	68.84 ± 0.16
			✓	71.38 ± 0.30	$\mathbf{63.23 \pm 0.35}$	$\mathbf{69.58 \pm 0.21}$
		✓	✓	$\mathbf{72.52 \pm 0.21}$	61.25 ± 0.27	$\mathbf{69.95 \pm 0.19}$
Russian	0			33.25 ± 0.18	$\mathbf{46.88 \pm 0.27}$	35.30 ± 0.12
		✓		$\mathbf{43.68 \pm 0.12}$	36.43 ± 0.24	$\mathbf{42.01 \pm 0.07}$
	8			37.44 ± 0.15	$\mathbf{40.79 \pm 0.32}$	38.07 ± 0.18
		✓		$\mathbf{46.19 \pm 0.25}$	32.49 ± 0.32	$\mathbf{42.60 \pm 0.18}$
			✓	38.18 ± 0.25	38.50 ± 0.17	38.24 ± 0.20
		✓	✓	45.38 ± 0.21	32.22 ± 0.05	41.96 ± 0.16

Table 5.3: Results of both methods on the non-English datasets. Checkmarks (✓) indicated if the corresponding method was used. The reported scores are the average between three random seeds, with indicated standard deviation. Bold values indicate the highest score each number of in-context examples.

5.3.2 TF-IDF GED

As it can be observed in Figure 5.3, using the SBERT to retrieve in-context examples is very close to randomly selecting them. This is expected, as semantical features are not particularly informative for the correction task. On the other hand, it is noticeable that the use of detection tags for example selection improves the results of the model. We observe that for all numbers of shots, TF-IDF GED beats or ties with SBERT and the random baseline. Observing the corresponding entries in Table 5.2 we observed that the the gains in $F_{0.5}$-score come from an increase in the precision of the model.

For the non-English dataset (Table 5.3) TF-IDF GED for example selection also improves the result of the baseline model. The biggest gain is observed in German, which can be explained by the GED model being trained in its equivalent training set. On the other hand, for Czech, the gains are not so expressive and for Russian the results are equivalent when considering the variance in the results.

It is interesting to notice that this experiment is performed using binary detection tags. Therefore, the example selection algorithm does not make use of any explicit information regarding the type of error in the sentence. In addition, when tags are predicted from a model trained with a different dataset more noise is present in the selection. This is most likely the reason why we do not observe substantial gains in Russian. Next, we show how there is room for improving the method by tackling the two aforementioned points.

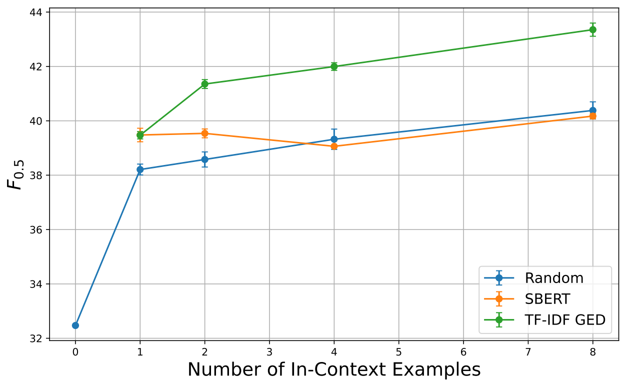

Figure 5.3: Comparison between the random selection, SBERT similarity, and TF-IDF GED on the W&I+LOCNESS development set.

Oracle Detection Tags

To demonstrate that this method can be more powerful with better GED systems we make use of two elements: ground-truth labels and finer detection granularity. Using ground-truth GED data we can get an upper bound of the proposed approach. Moreover, by utilizing more specific detection tags we observe how the model behaves when shown examples with even higher error similarity. For this experiment, we use detection labels with 2, 4 and 55 classes which correspond to operation, type and operation-type.

As Figure 5.4 demonstrates, the use of the ground-truth detection tags improves the scores across all numbers of in-context examples. This indicates that further developments in GED systems would automatically increase the quality of the TF-IDF GED method. Additionally, as we can see from the higher granularity methods, using GED tags that indicate the error operation or the specific error type increases the scores drastically. Thus, investigating ways to improve multiclass GED models would also benefit example selection.

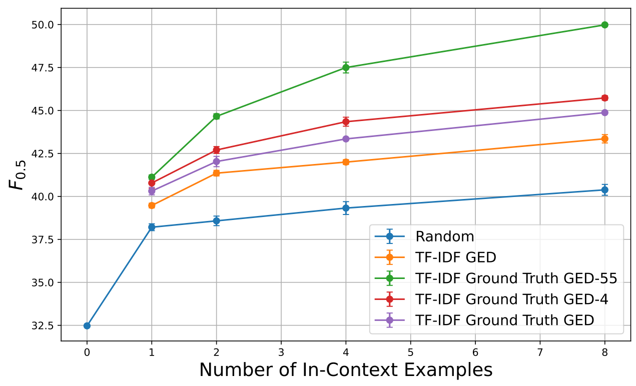

Figure 5.4: Comparison between the random selection, and TF-IDF GED with multiple levels of granularity and ground truth data on the W&I+LOCNESS development set.

5.3.3 Combining the Methods

Finally, we show the results when combining the methods to produce a more robust alternative for prompt-based GEC. In Table 5.2 we can observe the scores on the development set of W&I+LOCNESS. Importantly, the combination of the two methods improves the performance of both in isolation and considerably advances the baseline scores. The same result is observed for Czech, German and Russian in Table 5.3

Model	English	Czech	German	Russian
GECToR [50]	72.40	-	-	-
Náplava and Straka [49]	69.00	80.17	73.71	50.20
gT5 base [60]	60.20	71.88	69.21	26.24
gT5 xxl [60]	69.83	83.15	75.96	51.62
GPT3.5 [22]	47.00	-	63.50	-
Ours (GED Filter + TF-IDF GED)	55.1	75.95	70.91	42.11

Table 5.4: Results of the combination of filtering and TF-IDF GED in test sets in English, Czech, German and Russian. Dashes (-) indicate that the method was not available in the language. Models are divided into supervised and unsupervised, the first being supervised and the last unsupervised.

Table 5.4 shows the performance of combining our two proposed methods in the test sets of all languages. For this experiments we Llama2 7B for English and GPT3.5 for the other languages. Our method reduces the difference to supervised models in all four languages studied. For English, the results are still very far from supervised methods, but we beat the results of GPT-3.5 using Llama2 7B with our methods, a model that is considerably smaller. For Czech and Russian the results are better than gT5 base, a supervised GEC model. For German the model does not match supervised approaches, but gets a competitive score that is better than previous prompt-based works. This indicates that GED is a useful tool to improve prompt-based GEC, and more exploration should be done in this direction.

Chapter 6

Conclusions

In this work, we focused on better analyzing multilingual GED models and their applications for prompt-based GEC. On the evaluation of multilingual detection systems, we demonstrate that using a single multilingual model is preferred over a collection of monolingual models by analyzing in-domain, cross-dataset, and cross-lingual generalization. In all cases, the multilingual model performs better or in the worst case as well as the monolingual systems. By also taking into consideration the memory efficiency of only keeping track of a single model, it is clear that multilingual GED models are the best choice.

Once the effectiveness of GED models is assessed, we study how they can improve GEC performed through prompting LLMs. More specifically, we are interested in avoiding overcorrection and increasing the models' ability to generate corrections that better follow the style of the in-context examples.

To mitigate overcorrection we demonstrate that scores are improved by using a GED model to select which sentence should be passed to the LLM. By filtering out sentences that are predicted as correct by the detection system, we increase the scores and reduce LLM usage. The latter is very relevant as running the LLM is a computationally expensive process, given its size and auto-regressive nature. Therefore, GED filtering results in better performance in terms of evaluation metrics and faster inference.

For in-context example selection, we show that semantical similarity methods are as good as random selection for GEC. We demonstrate that by choosing in-context examples based on the error structure of the sentence better results are observed. The method, called TF-IDF GED, computes the TF-IDF vector of the detection tags n-grams, therefore it relies only on the error structure, ignoring the actual words of the sentence. Importantly, GED filtering and TF-IDF GED can be combined and together exhibit state-of-the-art performance in LLM-based GEC.

The advances proposed in this thesis point to a promising direction in combining GED models with LLM to generate correction. This hybrid approach, which makes use of a smaller specialized system, allows us to make use of the massive LLM only when necessary and improves it by generating more useful prompts. Continue on this line of work, many other solutions still need to be explored and evaluated. Now we show some interesting aspects for further investigation:

- Study of parameter efficient GED models that could be incorporated as a lightweight component of LLM prompt-based GEC
- Investigate multilingual GED models on finer levels of granularity that could be used to improve correction
- Expand the analysis performed on Llama2 7B to other model sizes and different LLM families

References

[1] Dimitris Alikaniotis and Vipul Raheja. The Unreasonable Effectiveness of Transformer Language Models in Grammatical Error Correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 127–133, Florence, Italy, August 2019. Association for Computational Linguistics.

[2] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Maitha Alhammadi, Mazzotta Daniele, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. The falcon series of language models: Towards open frontier models. 2023.

[3] Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. Parallel Iterative Edit Models for Local Sequence Transduction, May 2020. arXiv:1910.02893 [cs].

[4] Samuel Bell, Helen Yannakoudakis, and Marek Rei. Context is Key: Grammatical Error Detection with Contextual Word Representations. In Helen Yannakoudakis, Ekaterina Kochmar, Claudia Leacock, Nitin Madnani, Ildikó Pilán, and Torsten Zesch, editors, *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 103–115, Florence, Italy, August 2019. Association for Computational Linguistics.

[5] Adriane Boyd. Using Wikipedia Edits in Low Resource Grammatical Error Correction. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 79–84, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[6] Adriane Boyd, Jirka Hana, Lionel Nicolas, Detmar Meurers, Katrin Wisniewski, Andrea Abel, Karin Schöne, Barbora Štindlová, and Chiara Vettori. The MERLIN corpus: Learner language and the CEFR. In Nicoletta Calzolari, Khalid Choukri, Thierry

Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1281–1288, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA).

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.

[8] Christopher Bryant and Ted Briscoe. Language Model Based Grammatical Error Correction without Annotated Training Data. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 247–253, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[9] Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. The bea-2019 shared task on grammatical error correction. In *BEA@ACL*, 2019.

[10] Christopher Bryant, Mariano Felice, and Ted Briscoe. Automatic annotation and evaluation of error types for grammatical error correction. In *Annual Meeting of the Association for Computational Linguistics*, 2017.

[11] Christopher Bryant, Zheng Yuan, Muhammad Reza Qorib, Hannan Cao, Hwee Tou Ng, and Ted Briscoe. Grammatical error correction: A survey of the state of the art. (arXiv:2211.05166), March 2023. arXiv:2211.05166 [cs].

[12] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[13] Martin Chodorow and Claudia Leacock. An Unsupervised Method for Detecting Grammatical Errors. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*, 2000.

[14] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: pre-training text encoders as discriminators rather than generators. *CoRR*, abs/2003.10555, 2020.

[15] Davide Colla, Matteo Delsanto, and Elisa Di Nuovo. EliCoDe at MultiGED2023: fine-tuning XLM-RoBERTa for multilingual grammatical error detection. In *Proceedings of the 12th Workshop on NLP for Computer Assisted Language Learning*, pages 24–34, Tórshavn, Faroe Islands, May 2023. LiU Electronic Press.

[16] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116, 2019.

[17] Steven Coyne, Keisuke Sakaguchi, Diana Galvan-Sosa, Michael Zock, and Kentaro Inui. Analyzing the Performance of GPT-3.5 and GPT-4 in Grammatical Error Correction, May 2023. arXiv:2303.14342 [cs].

[18] Daniel Dahlmeier and Hwee Tou Ng. Better evaluation for grammatical error correction. In Eric Fosler-Lussier, Ellen Riloff, and Srinivas 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.

[19] Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. Building a large annotated corpus of learner english: The nus corpus of learner english. In *BEA@NAACL-HLT*, 2013.

[20] R. Dale, Ilya Anisimoff, and George Narroway. Hoo 2012: A report on the preposition and determiner error correction shared task. In *BEA@NAACL-HLT*, 2012.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

[22] Tao Fang, Shu Yang, Kaixin Lan, Derek F. Wong, Jinpeng Hu, Lidia S. Chao, and Yue Zhang. Is ChatGPT a Highly Fluent Grammatical Error Correction System? A Comprehensive Evaluation, April 2023. arXiv:2304.01746 [cs].

[23] Michael Gamon. High-Order Sequence Modeling for Language Learner Error Detection. In Joel Tetreault, Jill Burstein, and Claudia Leacock, editors, *Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications*,

pages 180–189, Portland, Oregon, June 2011. Association for Computational Linguistics.

[24] Sylviane Granger. The computer learner corpus: a versatile new source of data for sla research. 1998.

[25] Roman Grundkiewicz and Marcin Junczys-Dowmunt. *The WikEd Error Corpus: A Corpus of Corrective Wikipedia Edits and Its Application to Grammatical Error Correction*, volume 8686 of *Lecture Notes in Computer Science*, page 478–490. Springer International Publishing, Cham, 2014.

[26] Roman Grundkiewicz and Marcin Junczys-Dowmunt. Minimally-augmented grammatical error correction. In Wei Xu, Alan Ritter, Tim Baldwin, and Afshin Rahimi, editors, *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, page 357–363, Hong Kong, China, November 2019. Association for Computational Linguistics.

[27] Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. Neural Grammatical Error Correction Systems with Unsupervised Pre-training on Synthetic Data. In Helen Yannakoudakis, Ekaterina Kochmar, Claudia Leacock, Nitin Madnani, Ildikó Pilán, and Torsten Zesch, editors, *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263, Florence, Italy, August 2019. Association for Computational Linguistics.

[28] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[29] Kengo Hotate, Masahiro Kaneko, and Mamoru Komachi. Generating Diverse Corrections with Local Beam Search for Grammatical Error Correction. In Donia Scott, Nuria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2132–2137, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

[30] Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. A Nested Attention Neural Hybrid Model for Grammatical Error Correction. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 753–762, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[31] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task. In Marilyn Walker, Heng Ji, and Amanda 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.

[32] Masahiro Kaneko and Mamoru Komachi. Multi-Head Multi-Layer Attention to Deep Language Representations for Grammatical Error Detection, April 2019. arXiv:1904.07334 [cs].

[33] Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. Encoder-Decoder Models Can Benefit from Pre-trained Masked Language Models in Grammatical Error Correction. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4248–4254, Online, July 2020. Association for Computational Linguistics.

[34] Sudhanshu Kasewa, Pontus Stenetorp, and Sebastian Riedel. Wronging a Right: Generating Better Errors to Improve Grammatical Error Detection. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4977–4983, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[35] Satoru Katsumata and Mamoru Komachi. Stronger Baselines for Grammatical Error Correction Using a Pretrained Encoder-Decoder Model. In Kam-Fai Wong, Kevin Knight, and Hua Wu, editors, *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*, pages 827–832, Suzhou, China, December 2020. Association for Computational Linguistics.

[36] Olesya Kisselev, Anna Alsufieva, and Sandra Freels. Results 2012: Using flagship data to develop a russian learner corpus of academic writing. 65:79–105, 11 2012.

[37] Phuong Le-Hong, The Quyen Ngo, and Thi Minh Huyen Nguyen. Two neural models for multilingual grammatical error detection. In *Proceedings of the 12th Workshop on NLP for Computer Assisted Language Learning*, page 40–44, Tórshavn, Faroe Islands, May 2023. LiU Electronic Press.

[38] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdel rahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Annual Meeting of the Association for Computational Linguistics*, 2019.

[39] Yinghao Li, Xuebo Liu, Shuo Wang, Peiyuan Gong, Derek F. Wong, Yang Gao, Heyan Huang, and Min Zhang. TemplateGEC: Improving grammatical error correction with detection template. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6878–6892, Toronto, Canada, July 2023. Association for Computational Linguistics.

[40] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.

[41] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

[42] Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. Encode, Tag, Realize: High-Precision Text Editing, September 2019. arXiv:1909.01187 [cs].

[43] Behrang Mohit, Alla Rozovskaya, Nizar Habash, Wajdi Zaghouani, and Ossama Obeid. The first qalb shared task on automatic text correction for arabic. In *ANLP@EMNLP*, 2014.

[44] Jakub Náplava and Milan Straka. Grammatical error correction in low-resource scenarios. *ArXiv*, abs/1910.00353, 2019.

[45] Jakub Náplava, Milan Straka, Jana Straková, and Alexandr Rosen. Czech grammar error correction with a large and diverse corpus. *Transactions of the Association for Computational Linguistics*, 10:452–467, 2022.

[46] Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. The conll-2013 shared task on grammatical error correction. In *CoNLL Shared Task*, 2013.

[47] Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. The conll-2014 shared task on grammatical error correction. In *CoNLL Shared Task*, 2014.

[48] Diane Nicholls. The cambridge learner corpus-error coding and analysis. 1999.

[49] Jakub Náplava and Milan Straka. Grammatical Error Correction in Low-Resource Scenarios. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 346–356, Hong Kong, China, November 2019. Association for Computational Linguistics.

[50] Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. GECToR – Grammatical Error Correction: Tag, Not Rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online, July 2020. Association for Computational Linguistics.

[51] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[52] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[53] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[54] Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2019.

[55] Marek Rei. Semi-supervised Multitask Learning for Sequence Labeling, April 2017. arXiv:1704.07156 [cs].

[56] Marek Rei and Helen Yannakoudakis. Compositional Sequence Labeling Models for Error Detection in Learner Writing. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1181–1191, Berlin, Germany, August 2016. Association for Computational Linguistics.

[57] Marek Rei and Helen Yannakoudakis. Auxiliary Objectives for Neural Error Detection Models. In Joel Tetreault, Jill Burstein, Claudia Leacock, and Helen Yannakoudakis,

editors, *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 33–43, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[58] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Conference on Empirical Methods in Natural Language Processing*, 2019.

[59] Marc Reznicek, Anke Lüdeling, Cedric Krummes, Franziska Schwantuschke, Maik Walter, Karin Schmidt, and Hagen Hirschmann. Das falko-handbuch. korpusaufbau und annotationen. version 2.01, 09 2012.

[60] Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. A Simple Recipe for Multilingual Grammatical Error Correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 702–707, Online, August 2021. Association for Computational Linguistics.

[61] Sebastian Ruder, Ivan Vulić, and Anders Søgaard. Square one bias in NLP: Towards a multi-dimensional exploration of the research manifold. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2340–2354, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[62] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[63] Felix Stahlberg, Christopher Bryant, and Bill Byrne. Neural Grammatical Error Correction with Finite State Transducers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4033–4039, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[64] Felix Stahlberg and Shankar Kumar. Seq2Edits: Sequence Transduction Using Span-level Edit Operations, September 2020. arXiv:2009.11136 [cs].

[65] Milan Straka, Jakub Náplava, and Jana Straková. Character Transformations for Non-Autoregressive GEC Tagging, November 2021. arXiv:2111.09280 [cs].

[66] Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. Selective annotation makes language models better few-shot learners, 2022.

[67] Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. Tense and aspect error correction for esl learners using global context. In *Annual Meeting of the Association for Computational Linguistics*, 2012.

[68] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971, 2023.

[69] Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288, 2023.

[70] Harun Uz and Gülşen Eryiğit. Towards automatic grammatical error type classification for Turkish. In Elisa Bassignana, Matthias Lindemann, and Alban Petit, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 134–142, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics.

[71] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von 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.

[72] Elena Volodina, Christopher Bryant, Andrew Caines, Orphée De Clercq, Jennifer-Carmen Frey, Elizaveta Ershova, Alexandr Rosen, and Olga Vinogradova. MultiGED-2023 shared task at NLP4CALL: Multilingual Grammatical Error Detection. In *Proceedings of the 12th Workshop on NLP for Computer Assisted Language Learning*, pages 1–16, Tórshavn, Faroe Islands, May 2023. LiU Electronic Press.

[73] Elena Volodina, Lena Granstedt, Arild Matsson, Beáta Megyesi, Ildikó Pilán, Julia Prentice [Grosse], Dan Rosén, Lisa Rudebeck, Carl-Johan Schenström, Gunlög Sundberg, and Mats Wirén. The swell language learner corpus: From design to annotation. *The Northern European Journal of Language Technology*, 6:67–104, 12 2019.

[74] Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

[75] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. *ArXiv*, abs/2109.01652, 2021.

[76] Haoran Wu, Wenxuan Wang, Yuxuan Wan, Wenxiang Jiao, and Michael Lyu. ChatGPT or Grammarly? Evaluating ChatGPT on Grammatical Error Correction Benchmark, March 2023. arXiv:2303.13648 [cs].

[77] Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y. Ng. Neural Language Correction with Character-Based Attention, March 2016. arXiv:1603.09727 [cs].

[78] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. In *North American Chapter of the Association for Computational Linguistics*, 2020.

[79] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. (arXiv:2010.11934), March 2021. arXiv:2010.11934 [cs].

[80] Helen Yannakoudakis, Øistein E. Andersen, Ardeshir Geranpayeh, Ted Briscoe, and Diane Nicholls. Developing an automated writing placement system for esl learners. *Applied Measurement in Education*, 31:251 – 267, 2018.

[81] Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. A new dataset and method for automatically grading ESOL texts. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[82] Michihiro Yasunaga, Jure Leskovec, and Percy Liang. LM-Critic: Language Models for Unsupervised Grammatical Error Correction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7752–7763, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[83] Zheng Yuan and Ted Briscoe. Grammatical error correction using neural machine translation. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386, San Diego, California, June 2016. Association for Computational Linguistics.

[84] Zheng Yuan, Shiva Taslimipoor, Christopher Davis, and Christopher Bryant. Multi-Class Grammatical Error Detection for Correction: A Tale of Two Systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8722–8736, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[85] Wajdi Zaghouani, Nizar Habash, Houda Bouamor, Alla Rozovskaya, Behrang Mohit, Abeer Heider, and Kemal Oflazer. Correction annotation for non-native arabic texts: Guidelines and corpus. In *LAW@NAACL-HLT*, 2015.

[86] Wajdi Zaghouani, Behrang Mohit, Nizar Habash, Ossama Obeid, Nadi Tomeh, Alla Rozovskaya, Noura Farra, Sarah Alkuhlani, and Kemal Oflazer. Large scale arabic error annotation: Guidelines and framework. In *International Conference on Language Resources and Evaluation*, 2014.

[87] Yiming Zhang, Shi Feng, and Chenhao Tan. Active example selection for in-context learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9134–9148, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[88] Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. Improving Grammatical Error Correction via Pre-Training a Copy-Augmented Architecture with Unlabeled Data, June 2019. arXiv:1903.00138 [cs].