Ungrammatical-syntax-based In-context Example Selection for Grammatical Error Correction

Anonymous NAACL submission

Abstract

In the era of large language models (LLMs), in-context learning (ICL) stands out as an effective prompting strategy that explores LLMs' potency across various tasks. However, applying LLMs to grammatical error correction (GEC) is still a challenging task. In this paper, we propose a novel ungrammatical-syntax-based in-context example selection strategy for GEC. Specifically, we measure similarity of texts based on their syntactic structure with diverse algorithms, and identify optimal ICL examples sharing the most similar ill-formed syntax to the test sample. Additionally, we carry out a two-stage process to further improve the quality of selection results. On benchmark English GEC datasets, empirical results show that our proposed ungrammatical-syntax-based strategies outperform commonly-used word-matching methods with multiple LLMs. This indicates that for a syntax-oriented task like GEC, paying more attention to syntactic information can effectively boost LLMs' performance. Our code will be publicly available after the publication of this paper.

1 Introduction

Recently, large language models (LLMs) have shown awesome power in many areas and ended the contest on many tasks (Chowdhery et al., 2023; Bubeck et al., 2023; Touvron et al., 2023). Unfortunately for LLMs, grammatical error correction (GEC), which aims at automatically correcting grammatical errors in erroneous text (Bryant et al., 2022), is still a challenging task where they cannot beat conventional models yet. Fang et al. (2023b) and Loem et al. (2023) explore the performance of LLMs on GEC, demonstrating mainstream LLMs lag over 10 points behind the state-of-the-art result. Therefore, it is significant to explore new strategies to further improve the power of LLMs on GEC.

In the era of LLMs, in-context learning (ICL) has achieved impressive results on many tasks (Dong et al., 2022; Min et al., 2022). In ICL, several in-context examples are presented to LLMs as demonstrations before the input test sample in order to make LLMs aware of the requirement and output format of the specific task, thereby enhancing LLMs' performance during the subsequent generation process. Since the quality of in-context examples plays a crucial role under the few-shot setting, some special-designed strategies of example selection and permutation have been proposed (Agrawal et al., 2023; Li et al., 2023a).

To the best of our knowledge, most works on ICL example selection focus on superficial word matching like BM25 (Robertson et al., 1994), without considering syntactic information. However, GEC aims to correct grammatical errors and is a typical syntax-oriented task. In GEC, common errors can be classified into four types: $misuse$, $missing$, $redundancy$ and $word\ order$ (Bryant et al., 2017; Zhang et al., 2022a), and the last three of which are closely related to syntactic structure. That is, the $missing$, $redundancy$ or $disorder$ of text constituents may lead to ill-formed syntax (Zhang et al., 2022b), suggesting the important role syntax plays in GEC. Hence, selecting in-context examples based on syntactic structure is likely to benefit LLMs more than conventional word-matching-based approaches.

Comparing with other natural language processing (NLP) tasks, syntactic similarity of text is less-studied. Previous works have leveraged the similarity of dependency trees to help multi-document summarization (Özateş et al., 2016) and semantic textual similarity (Le et al., 2018). To compute syntactic similarity, several effective algorithms of tree similarity have been proposed. Tree Kernel is a typical one, which counts the shared sub-structures of two trees to measure their similarity (Collins and Duffy, 2002; Vishwanathan et al., 2004; Moschitti, 2006). Polynomial Distance is another handy one, which converts syntactic trees into polynomials and

	Source (Erroneous Sentence)	Target (Corrected Sentence)
Input	No smoking in *the* public places.	No smoking in public places.
BM25	I am writing to complain about the suggested *bar* on smoking in public areas.	I am writing to complain about the suggested *ban* on smoking in public areas.
Poly.	No future for *the* public transport?	No future for public transport?

Table 1: An example comparing the selection results of BM25 and polynomial distance ("Poly." in the table).

then computes the distances (Liu et al., 2022).

In this paper, we propose a new ICL example selection strategy for GEC, by computing similarities of syntactic trees on ungrammatical sentences. Specially, we apply the syntactic similarity algorithms (Tree Kernel and Polynomial Distance) to dependency trees generated by a GEC-oriented parser (GOPar) proposed by Zhang et al. (2022b), which is more reliable and provides error information when parsing ungrammatical sentences. Moreover, we carry out a two-stage process. In the first stage, namely *selection*, a fast and general method like BM25 is applied to filter out most of the irrelevant instances from the training data and obtain a much smaller candidate set. In the second stage, namely *ranking*, the more powerful syntax-based method is implemented to find out the best k instances as the final in-context examples.

To give a quick view of the superiority of our method, Table 1 shows an example illustrating the difference between BM25 selection and our ungrammatical-syntax-based method with polynoimal distance selection. BM25 only selects examples with similar words while Polynomial Distance is able to select those with similar grammatical errors, which will benefit more the GEC task.

We conduct experiments on two English GEC datasets, BEA-2019 (Bryant et al., 2019) and CoNLL-2014 (Ng et al., 2014). According to experimental results, Polynomial Distance and its weighted version achieve competitive results even under the single-stage setting, improving the performance by around 3 points and 2 points on BEA-19 and CoNLL-14 respectively. With the help of our two-stage selection, Tree Kernel gets its power unlocked and Polynomial Distance also benefits, leading to a further 1-point and 0.4-point improvement on BEA-19 and CoNLL-14 respectively. Overall, our ungrammatical-syntax-based in-context example selection methods secure the best results under all settings, outperforming conventional baselines by a margin of nearly 3 $F_{0.5}$ points on average.

Our contributions can be summarized as follows:

- We propose a novel ICL example selection method based on ungrammatical syntactic similarity to improve LLMs' performance on GEC. To the best of our knowledge, this is the first time that syntactic structure knowledge is introduced to ICL example selection for GEC.
- We explore a two-stage selection strategy on GEC, where superficial word-similarity-based methods are used in the first stage and deep syntax-similarity-based ones are used in the second stage. It further improves LLMs' performance and achieves competitive results.
- We want to re-draw the NLP community's attention to the significance of syntactic information. In this work, we show that syntax-related knowledge helps LLMs correct grammatical errors better. We believe our methods can be smoothly transferred to many other syntax-related tasks like machine translation (MT), information extraction (IE), etc.

2 Related Work

2.1 Grammatical Error Correction

In the past few years, the GEC task has been dominated by sequence-to-sequence machine translation models (Junczys-Dowmunt et al., 2018; Rothe et al., 2021) and sequence-to-edit tagging models (Omelianchuk et al., 2020; Tarnavskyi et al., 2022), both based on Transformer (Vaswani et al., 2017).

Nowadays, with the finalization of mainstream models, further explorations on GEC mainly focus on two aspects. For one thing, injecting all kinds of additional knowledge into GEC models has been proved helpful. The additional knowledge can be part-of-speech (POS) (Wu and Wu, 2022), syntax tree (Zhang et al., 2022b), speech representation (Fang et al., 2023a), abstract meaning representation (AMR) (Cao and Zhao, 2023), error type (Yang et al., 2023), etc. For another, multi-stage strategies help refine models' predictions. The multi-stage workflow can be permutation & decoding

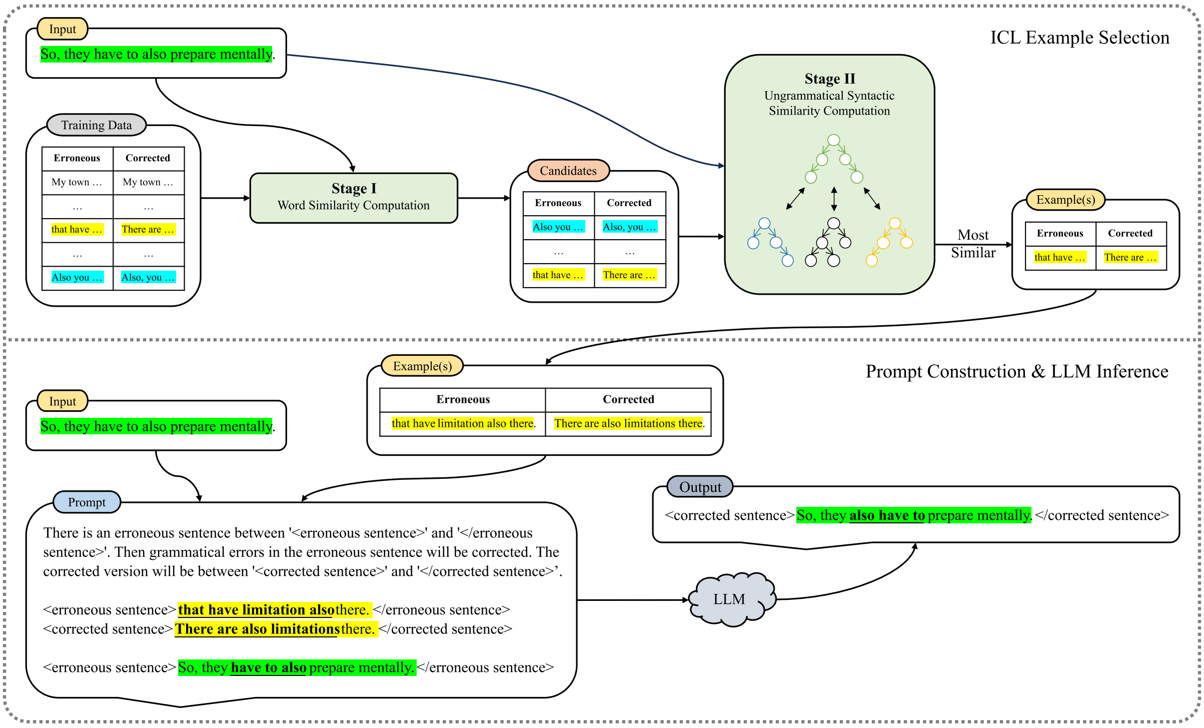


Figure 1: Our two-stage selection and ICL workflow. For each input test sample, Stage **I** computes word similarities with BM25 or BERT representation between the input and all training data and select the top-1000 as candidates. Then, Stage **II** computes ungrammatical syntactic similarities with tree kernel or polynomial distance between the input and candidates to select the most similar k example(s). After that, we concatenate the input after the k examples to construct the prompt for LLM inference. In the end, the LLM outputs the final result.

(Yakovlev et al., 2023), detection & correction (Li et al., 2023b), re-ranking (Zhang et al., 2023a), etc.

With the rising of powerful large language models (LLMs), some works have begun exploring their performance on GEC (Loem et al., 2023; Fang et al., 2023b), showing that LLMs cannot beat conventional models on GEC yet.

2.2 Syntactic Similarity

In computational linguistics (CL), previous works compared syntax trees of different languages to measure their similarities (Oya, 2020; Liu et al., 2022). In NLP, most works on text similarity focus on the semantic perspective (Gomaa et al., 2013, Reimers and Gurevych, 2019; Chandrasekaran and Mago, 2021), syntactic similarity of text is less-studied. Özateş et al. (2016) used similarity of dependency trees to help multi-document summarization. Le et al. (2018) proposed ACV-tree (Attention Constituency Vector-tree), which combines word weight, word representation and constituency tree, to help the task of semantic textual similarity.

Syntactic similarity is usually represented by similarity between syntax trees. Tree similarity can be measured by Edit Distance (de Castro Reis et al., 2004), Polynomial Distance (Liu et al., 2022), Subset Tree Kernel (SSTK) (Collins and Duffy, 2002), SubTree Kernel (STK) (Vishwanathan et al., 2004), Patial Tree Kernel (PTK) (Moschitti, 2006), etc.

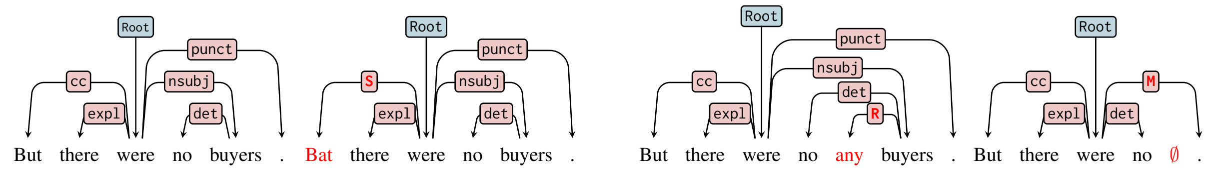
2.3 Large Language Models and In-context Learning

In recent years, LLMs have shown their awesome power in many areas (Brown et al., 2020; Chowdhery et al., 2023). Due to the limitation of computing resources, the focus of research on LLMs turns to the inference stage, trying to exploit the potency of LLMs with inference-only strategies.

ICL is a successful inference strategy that can make LLMs perform as well as fine-tuned models on many tasks (Brown et al., 2020; Von Oswald et al., 2023), where several in-context examples are given to LLMs as demonstrations before the actual test sample. Instead of randomly sampling examples from the training set, recent works have boosted the performance of ICL by selecting in-context examples using strategies based on similarity (Agrawal et al., 2023; Li et al., 2023a) or diversity (Zhang et al., 2023b).

LLaMA-2	GPT-3.5
There is an erroneous sentence between '<erroneous sentence>' and '</erroneous sentence>'. Then grammatical errors in the erroneous sentence will be corrected. The corrected version will be between '<corrected sentence>' and '</corrected sentence>'. <erroneous sentence> {e_1} </erroneous sentence> <corrected sentence> {c_1} </corrected sentence> <erroneous sentence> {e_2} </erroneous sentence> <corrected sentence> {c_2} </corrected sentence> <erroneous sentence> {e_3} </erroneous sentence> <corrected sentence> {c_3} </corrected sentence> <erroneous sentence> {e_4} </erroneous sentence> <corrected sentence> {c_4} </corrected sentence> <erroneous sentence> {e_{test}} </erroneous sentence> <corrected sentence>	"system": You are a grammar correction assistant. The user will give you a sentence with grammatical errors (between '<erroneous sentence>' and '</erroneous sentence>'). You need to correct the sentence (between '<corrected sentence>' and '</corrected sentence>'). Requirements: 1. Make as few changes as possible. 2. Make sure the sentence has the same meaning as the original sentence. 3. If there is no error, just output 'No errors found'. "user": <erroneous sentence> {e_1} </erroneous sentence> "assistant": <corrected sentence> {c_1} </corrected sentence> "user": <erroneous sentence> {e_2} </erroneous sentence> "assistant": <corrected sentence> {c_2} </corrected sentence> "user": <erroneous sentence> {e_3} </erroneous sentence> "assistant": <corrected sentence> {c_3} </corrected sentence> "user": <erroneous sentence> {e_4} </erroneous sentence> "assistant": <corrected sentence> {c_4} </corrected sentence> "user": <erroneous sentence> {e_{test}} </erroneous sentence>

Table 2: Prompts we use. e and c denote the erroneous and corrected sentences of in-context examples or test samples respectively.



(a) The correct sentence. (b) Substituted errors. (c) Redundant errors. (d) Missing errors.

Figure 2: Original illustration of GOPar from Zhang et al. (2022b). ∅ denotes the missing word.

Besides normal ICL, Chain-of-thought (CoT) (Wei et al., 2022; Kojima et al., 2022) is another effective inference strategy in current favor, where LLMs are prompted to think step by step and answer with intermediate rationales.

3 Methodology

3.1 In-context Learning Workflow for GEC

Based on LLMs, our ungrammatical-syntax-based example selection and few-shot ICL workflow is illustrated in Figure 1. Specially, when faced with a test sample, we search through the training data to find the best example(s) for in-context learning. Then, both the source (erroneous) and the target (corrected) sentences of the example(s) are inserted into the prompt as demonstrations, with the test sample concatenated at the end. In this way, LLMs can learn the GEC task from the demonstrations and perform better correction on the test sample. In this framework, a set of high-quality in-context examples are crucial to lead LLMs to a better performance. Prompts used in this work are shown in Table 2.

3.2 Syntax Parser for Ungrammatical Sentences

Unlike most NLP tasks, which take correct sentences as input, the GEC task considers erroneous text as input. This gives rise to an issue that mainstream parsers may fail to obtain the expected dependency tree for the erroneous text.

To solve this problem, Zhang et al. (2022b) built a tailored GEC-Oriented dependency Parser (GOPar) based on the parallel GEC training data, which is much more reliable when handling ungrammatical sentences than conventional parsers. Concretely, GOPar sets "S" (Substituted), "R" (Redundant) or "M" (Missing) labels to deal with different kinds of grammatical errors in the sentence, which inject additional information of errors into the dependency tree. Figure 2 shows the original illustration of GOPar from Zhang et al. (2022b).

Most previous works computing syntactic similarity base on grammatical sentences with standard parsing trees (Özateş et al., 2016; Oya, 2020). However, in GEC, we only have the ungrammatical source sentences, on which conventional parsers may perform poorly. So we apply the algorithms of tree similarity on the parsing results of GOPar, to compute syntactic similarities between test sample and training instances.

3.3 Syntactic Similarity with Tree Kernel

We follow the unified Tree Kernel method proposed by (Moschitti, 2006), which can compute kernels of subset trees defined by Collins and Duffy (2002), subtrees defined by Vishwanathan et al. (2004) and partial trees defined in their own work.

For brevity, we imitate the algorithm described in Le et al. (2018) and design the following algorithm (shown in Algorithm 1) to implement a simple version of Tree Kernel.

Algorithm 1 Similarity with Tree Kernel

```
procedure COMPSIM(N1, N2)
    K ← 0
    for each node n_i in N1 do
        for each node n_j in N2 do
            if n1.label = n2.label then
                if n1 and n2 are both leaves then
                    K ← K + 1
                else if n1 and n2 are both non-leaves then
                    K ← K + COMPSIM(n1, n2)
                end if
            end if
        end for
    end for
    K ← K/(N1.size × N2.size)
    return K
end procedure
```

For two trees T_1 and T_2, we conduct COMPSIM between their root nodes N_1 and N_2 to get a similarity score.

3.4 Syntactic Similarity with Polynomial Distance

Liu et al. (2022) converted trees into polynomials and took the distances between polynomials as tree distances to measure syntactic similarities of dependency trees.

Given the number of dependency labels d, the dependency trees will be represented into polynomials recursively on two variable set: $X = \{x_1, x_2 ... x_d\}$ and $Y = \{y_1, y_2, ... y_d\}$. In the dependency tree, for each leaf n^l with label l, the corresponding polynomial is $P(n^l, X, Y) = x_l$. Then, for each non-leaf m_l with label l, the corresponding polynomial is $P(m^l, X, Y) = y_l + \prod_{i=1}^{k} P(n_i, X, Y)$, where $n_1, ..., n_k$ are all child nodes of m_l. In this way, the polynomial of the root node is regarded as the polynomial representation of a tree.

To compute similarity more conveniently, for each term $cx_1^{e_{x_1}} x_2^{e_{x_2}} ... x_d^{e_{x_d}} y_1^{e_{y_1}} y_2^{e_{y_2}} ... y_d^{e_{y_d}}$ in the dependency polynomial, we write it as a term vector with $2d+1$ entries:

$$t = [e_{x_1}, e_{x_2}, ..., e_{x_d}, e_{y_1}, e_{y_2}, ..., e_{y_d}, c],$$

where each entry represents the exponent of the corresponding variable. In this way, a polynomial P can be written as a set of term vectors $\mathcal{V}_P$. Then, we compute the distance between two polynomials as:

$$d(P, Q) = \frac{\sum_{s \in \mathcal{V}_P} \min_{t \in \mathcal{V}_Q} \| s - t \|_1 + \sum_{t \in \mathcal{V}_Q} \min_{s \in \mathcal{V}_P} \| s - t \|_1}{| \mathcal{V}_P | + | \mathcal{V}_Q |}, \tag{1}$$

where $\| s - t \|_1$ denotes the Manhattan distance (Craw, 2017) between term vector s and t.

Weighting Ungrammatical Nodes We hypothesize that LLMs benefit more from similar grammatical errors, and error nodes with similar neighboring syntactic structure lead to similar error patterns. Therefore, assigning higher weights to ungrammatical nodes can select examples with error patterns closer to the test sample. Hence, besides the original algorithm, we also explore a weighted version. When computing the Manhattan distance between two term vectors, we assign a higher weight to entries corresponding to labels with error information ("S", "R" and "M"). In our experiment, as a preliminary attempt, we set the weight to 2.

3.5 Two-stage Selection

In previous works, a two-stage select-then-rank strategy performs well in in-context learning (Wu et al., 2023; Agrawal et al., 2023). To be specific, a fast and general method is used to filter out most of the not-so-relevant instances from training data and get a much smaller candidate set with high quality, which is called *selection*. After that, a specific and powerful method is used to rank the instances in the candidate set and obtain the top-k best training instances, which is called *ranking*. Motivated by this, we also design a two-stage sample selection mechanism for GEC.

Stage 1: BM25/BERT Selection First, we explore *selection* with BM25 or BERT representation to obtain candidate examples, and the size of candidate set is 1000 in our experiment.

BM25 (Robertson et al., 1994) is a widely-used retrieval algorithm based on term frequency, inverse document frequency and length normalization. Many recent works regard BM25 as a strong baseline for in-context example selection (Agrawal et al., 2023; Li et al., 2023a). In our work, we take the input test sample as the query and source sentences of all training data as the document.

BERT Representation Li et al. (2023a) make use of SentenceBERT (Reimers and Gurevych,

I	II	BEA-2019									CoNLL-2014								
		LlaMA-2-7B			LlaMA-2-13B			GPT-3.5-turbo			LlaMA-2-7B			LlaMA-2-13B			GPT-3.5-turbo		
		P	**R**	**F$_{0.5}$**	**P**	**R**	**F$_{0.5}$**	**P**	**R**	**F$_{0.5}$**	**P**	**R**	**F$_{0.5}$**	**P**	**R**	**F$_{0.5}$**	**P**	**R**	**F$_{0.5}$**
-	Rand.	50.1	57.7	51.5	49.0	61.2	51.0	47.0	70.4	50.3	59.4	48.8	56.9	58.6	51.3	57.0	56.5	59.9	57.1
	BM25	50.9	58.2	52.2	51.6	61.1	53.3	46.8	69.6	50.1	59.7	47.7	56.8	59.3	50.1	57.2	56.6	60.8	57.4
	BERT	50.7	56.8	51.8	51.0	61.2	52.8	47.6	70.0	50.9	58.6	45.4	55.4	60.1	52.0	58.3	56.0	60.8	56.9
	T. K.	50.0	57.0	51.2	52.5	59.0	53.6	47.2	69.8	50.5	57.9	47.5	55.5	61.8	48.0	58.5	57.3	60.3	57.9
	Poly.	53.1	57.9	54.0	52.9	60.2	54.3	49.5	70.0	52.6	59.5	49.5	57.2	61.7	51.8	59.4	58.2	59.9	58.6
	W. Poly.	53.2	58.2	**54.2**	53.4	60.5	**54.7**	50.3	69.6	**53.2**	60.1	49.2	**57.5**	61.6	52.3	**59.5**	58.4	60.5	**58.8**
BM25	T. K.	55.1	55.9	**55.2**	54.9	58.7	**55.6**	49.7	69.3	**52.7**	62.2	45.7	58.0	61.9	47.3	58.3	58.3	59.7	**58.6**
	Poly.	51.2	57.1	52.3	50.9	59.8	52.5	48.8	69.5	51.9	62.1	47.7	**58.6**	60.9	49.8	58.3	57.2	59.7	57.7
	W. Poly.	54.4	57.4	55.0	54.0	59.7	55.0	49.3	69.8	52.4	61.4	47.7	58.1	60.8	50.4	**58.4**	57.6	60.4	58.1
BERT	T. K.	53.6	56.0	54.1	53.7	59.3	54.7	50.0	69.7	**53.0**	60.7	46.3	57.1	60.8	49.9	**58.3**	57.6	59.2	57.9
	Poly.	53.3	57.2	54.0	53.8	60.4	55.0	49.0	69.5	52.1	60.5	47.6	57.4	59.8	50.8	57.8	57.6	60.7	**58.2**
	W. Poly.	53.8	57.4	**54.5**	54.2	60.7	**55.4**	49.9	69.7	52.9	61.0	48.3	**57.9**	59.8	51.5	57.9	57.3	60.5	57.9

Table 4: Experimental results under the few-shot setting with 4 examples. **I** and **II** denote the first (*selection*) and second (*ranking*) stage of the two-stage selection respectively. "-" means the Stage **I** is absent and these are single-stage models. "Rand.", "T. K.", "Poly." and "W. Poly." refer to "Random", "Tree Kernel" "Polynomial Distance" and "Weighted Polynomial Distance", respectively. The dashed line separates results of conventional baselines and our proposed methods: the former on the upper side and the latter on the lower side. The best F$_{0.5}$ scores of each group are displayed in **bold**, and the best F$_{0.5}$ scores of all settings are displayed in **underlined bold**.

2019) to get sentence representations and then compared similarities of sentences. For brevity, we adopt the more frequently-used BERT (Devlin et al., 2019) instead. In our work, we take the BERT representation of the `[CLS]` token as the representation of the sentence. Then we compute the cosine similarities between the representations of the input test sentence and all source sentences in the training data.

For comparison, we also experiment on single-stage BM25 and BERT representation selection, which serve as baselines in Section 4.

Stage 2: Ungrammatical-syntax-based Ranking Further, we employ *ranking* via syntactic similarity computing with Tree Kernel or Polynomial Distance, to obtain the best k matching examples from the candidate set.

4 Experimental Results

4.1 Datasets and Evaluation Metrics

Dataset	#Sentences	%Error	Usage
W&I+LOCNESS	34,308	66	Demonstration
BEA-19-Test	4,477	-	Testing
CoNLL-14-Test	1,312	72	Testing

Table 3: Statistics of GEC datasets used in this work. **#Sentences** refers to the number of sentences. **%Error** refers to the percentage of erroneous sentences.

We carry out experiments on English GEC datasets. Since no model training is involved, most large-scale GEC data is unnecessary, but the data quality matters for example selection. Thus in this work, we only use the relatively small but high-quality Write&Improve+LOCNESS (W&I+LOCNESS) (Bryant et al., 2019) as the training data.

For evaluation, we report P (Precision), R (Recall) and F$_{0.5}$ results on BEA-19 test set (Bryant et al., 2019) evaluated by ERRANT (Bryant et al., 2017) and on CoNLL-14 test set (Ng et al., 2014) evaluated by M2Scorer (Dahlmeier and Ng, 2012). We primarily compare the F$_{0.5}$ among different methods, which shows the comprehensive performance of models on GEC.

Statistics of datasets mentioned above are shown in Table 3.

4.2 Large Language Models

We use two mainstream LLM series: LLaMA-2 (Touvron et al., 2023) and GPT-3.5 (OpenAI, 2023) for experiment.

For LLaMA-2, we use `llama-2-7b-chat` and `llama-2-13b-chat` with 7B and 13B parameters respectively. For GPT-3.5, we use the official `gpt-3.5-turbo` API for inference.

For the sake of reproductivity, we turn off the sampling and set the temperature to zero for all these models we use.

4.3 Results

The experimental results are shown in Table 4. With different LLMs and on both datasets, our ungrammatical-syntax-based selection strategy obviously outperforms traditional methods (BM25 and BERT representation). On BEA-2019 data, the method with first BM25 selection and then Tree Kernel ranking improves the performance by 3.7, 4.6 and 2.4 $F_{0.5}$ points, using `llama-2-7b-chat`, `llama-2-13b-chat` and `gpt-3.5` respectively.

Performance of Tree Kernel When applied as a single-stage method, the Tree Kernel similarity performs poorly and even achieves a lower $F_{0.5}$ score than conventional baselines. However, with the help of a preliminary *selection* stage, it improves by a margin of about 2 to 3 percentage points, and even achieves the highest $F_{0.5}$ score on BEA-2019 data with LLaMA-2.

Performance of Polynomial Distance Different from Tree Kernel, Polynomial Distance performs fairly well even without a preliminary *selection*. Among those single-stage approaches, both polynomial-based methods outperform traditional baselines by an average margin of 2 to 3 percentage points in all cases, which indicates the superiority of syntactic similarity on GEC. The weighted version, with a higher weight on labels with error tags, brings a slight improvement in most cases, which shows the effectiveness of error information in GOPar-based dependency trees.

Performance of Two-stage Selection As for Tree Kernel, the two-stage selection strategy consistently boosts performance, whether using BM25 or BERT representation as the preliminary selection approach. But for Polynomial Distance, the two-stage selection fails to improve performance in most cases, and we leave it for future research.

5 Model Analysis

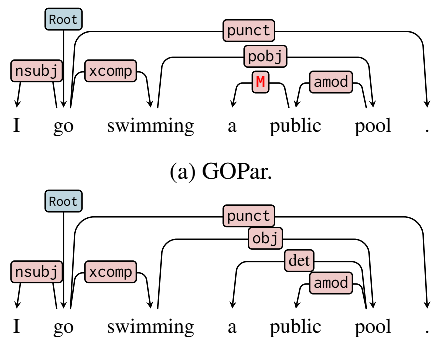
5.1 Experiments with Different Numbers of Prompt Examples

To explore the consistency and robustness of our methods, we conduct 1-shot, 2-shot, 4-shot and 8-shot experiments on `llama-2-7b-chat`. The results on BEA-2019 are shown in Table 5, and results on CoNLL-2014 are listed in Appendix A to save space.

When there is only one example, the model performs relatively poor. When the number of examples comes to two, the performance improves significantly. Then, further increasing the number of examples brings a slight but consistent performance gain.

When the number of examples is small, the superiority of syntax-based methods compared with those conventional is evident. When the number of examples increases, conventional baselines improve a lot while syntax-based methods gain relatively less, which shows a marginal benefit. But syntax-based methods always secure the highest score, indicating the consistency of their advantages. Especially, the single-stage Polynomial Distance and the two-stage BM25 plus Tree Kernel using 2 examples achieve very competitive results with traditional selection methods using 8 examples.

5.2 Ungrammatical Parser or Standard Parser?



(a) GOPar.

(b) Stanford Parser.

Figure 3: An example of parsing tree by GOPar and Stanford Parser.

To explore the affect of different parsers on model performance, we also experiment with Stanford Parser (Dozat and Manning, 2017), which is a widely-used conventional parser. For a clear demonstration, an example is illustrated in Figure 3 to show the different parsing results of GOPar and Stanford Parser.

The experimental results comparing GOPar and Stanford Parser on BEA-2019 test set are shown in Table 6. Here, we adopt `llama-2-7b-chat` as the LLM and Tree Kernel as the ranking method.

Without using the two-stage selection, Stanford Parser performs slightly worse than GOPar. With the two-stage selection, GOPar gains more improvement than Stanford Parser and outperforms it by a margin of more than 2 points. This indicates

I	II	1-shot			2-shot			4-shot			8-shot		
		P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
-	Rand.	47.3	29.8	42.3	49.6	50.9	49.8	50.1	57.7	51.5	52.2	58.8	53.4
	BM25	48.4	35.8	**45.2**	50.4	53.1	50.9	50.9	58.2	52.2	52.5	59.0	53.7
	BERT	47.3	33.8	43.8	50.0	51.4	50.2	50.7	56.8	51.8	53.6	59.3	54.6
	T. K.	47.1	27.4	41.2	49.0	53.2	49.8	50.0	57.0	51.2	53.6	55.9	54.0
	Poly.	50.1	31.5	44.8	53.9	51.9	**53.5**	53.1	57.9	54.0	54.3	58.3	**55.1**
	W. Poly.	50.4	31.5	45.0	52.7	51.5	52.4	53.2	58.2	**54.2**	53.3	58.0	54.2
BM25	T. K.	51.7	37.5	**48.1**	53.3	53.8	**53.4**	55.1	55.9	**55.2**	57.2	55.6	**56.9**
	Poly.	51.3	36.6	47.5	52.9	54.5	53.2	51.2	57.1	52.3	55.5	56.9	55.8
	W. Poly.	51.1	36.6	47.4	52.8	54.7	53.2	54.4	57.4	55.0	56.3	57.0	56.4
BERT	T. K.	50.7	35.6	46.8	53.3	52.4	**53.1**	53.6	56.0	54.1	57.1	57.0	**57.1**
	Poly.	50.9	35.5	**46.9**	52.1	53.4	52.4	53.3	57.2	54.0	55.5	58.2	56.1
	W. Poly.	50.6	35.7	46.7	52.1	53.8	52.4	53.8	57.4	**54.5**	56.5	57.8	56.7

Table 5: Results of different numbers of shots on BEA-19 test set.

	Source (Erroneous Sentence)	Target (Corrected Sentence)
Input	So, they *have to also* prepare mentally.	So, they *also have to* prepare mentally.
BM25	Also you can see how they prepare your food in front of you.	Also, you can see how they prepare your food in front of you.
T. K.	Nowadays people *get around constantly*.	Nowadays, people *are constantly on the move*.
BM25 + T. K.	*that have limitation also* there.	*There are also limitations* there.

Table 7: A one-shot example showing the tree kernel method benefiting from the two-stage selection.

I	II	GOPar			Stanford Parser		
		P	R	$F_{0.5}$	P	R	$F_{0.5}$
-	T. K.	50.0	57.0	51.2	49.6	56.4	50.8
BM25		55.1	55.9	**55.2**	51.8	56.2	**52.7**
BERT		53.6	56.0	54.1	50.6	57.2	51.8

Table 6: Results on BEA-2019 test set with 4 examples, using GOPar and Stanford Parser respectively.

GOPar is more suitable for GEC, and its superiority lies in two aspects. First, it performs more robust on ungrammatical sentences (e.g., it correctly recognizes the prepositional object "pool" in the sentence shown in Figure 3 while Stanford Parser fails to). Second, it provides extra information about the grammatical errors (e.g., the *Missing* error in Figure 3).

5.3 Effect of Two-stage Selection

In order to find out how the two-stage strategy benefits the Tree Kernel method, we conduct a case study and compare three selection settings: BM25 only ("BM25"), Tree Kernel only ("T. K.") and Tree Kernel after a BM25 *selection* ("BM25 + T. K.").

In the example shown in Table 7, the input sentence is ungrammatical in word order. "BM25" selects a sentence with a punctuation missing error that is similar to the input sample in words ("also", "they" and "prepare"). "T. K." selects a sentence with an improper expression "get around constantly" which is similar to "prepare mentally" in syntactic structure but has little to do with the grammatical errors. "BM25 + T. K." selects a sentence that is similar to the input sample both in word occurrences ("also" and "have") and in error form (improper word order).

Since similar words are more likely to form similar errors, with the help of a preliminary selection, Tree Kernel can select from a more relative candidate set, leading to a better example selection involving both word and syntactic similarity in erroneous constituents. Moreover, it also shows the disadvantage of conventional selection method BM25 on GEC, which cannot effectively select examples similar in syntax.

6 Conclusion

In this work, we make use of two conventional tree-based syntactic similarity algorithms and the select-then-rank two-stage framework to select in-context examples for the GEC task. Empirical results show that our syntax-based in-context example selection method is effective for GEC. We call on the NLP community to pay more attention to the help of syntactic information for many other syntax-related tasks besides GEC.

Limitations

First, we only experiment on English datasets. The performance of our method on other languages requires further exploration. Second, besides dependency tree, constituent tree is also worth trying. However, unfortunately, we do not have access to GEC-oriented constituent trees (Zhang and Li, 2022) at the time of writing this paper. Third, many previous outstanding methods of both in-context example selection and tree similarity computation have not been explored in our work. Fourth, due to limited time, we do not explore the effect of the size of candidate set after the *selection* stage and the choice of weight of ungrammatical nodes in the Polynomial Distance method. There may exist a better size than the values we use in our experiments. Last, except for the Stanford Parser, our experiments do not split sentences on both training and test data. Some instances in GEC datasets contains more than one sentence. Directly parsing these instances without splitting sentences may hurt the parsing performance and lead to unreliable results.

Ethics Statement

Use of Scientific Artifacts. We make use of GOPar provided by Zhang et al. (2022b), which is publicly available based on the MIT license [1].

About Computational Budget. Computation time is shown in Table 8.

Method	Time
BM25	440
BERT	4500
Tree Kernel	3600
Polynomial Distance	3200

Table 8: Computation time of different methods on BEA-19 test set, all in seconds. BERT runs on an NVIDIA GeForce RTX 2080 Ti and the other three run on an Intel® Xeon® Gold 5218 CPU.

About Reproducibility. All the experiments are completely reproducible since we disable sampling and set the temperature to zero for all LLMs we use, as discussed in Section 4.2.

[1] https://github.com/HillZhang1999/SynGEC

References

Sweta Agrawal, Chunting Zhou, Mike Lewis, Luke Zettlemoyer, and Marjan Ghazvininejad. 2023. In-context examples selection for machine translation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8857–8873, Toronto, Canada. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

Christopher Bryant, Zheng Yuan, Muhammad Reza Qorib, Hannan Cao, Hwee Tou Ng, and Ted Briscoe. 2022. Grammatical error correction: A survey of the state of the art. *Computational Linguistics*, pages 1–59.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Hejing Cao and Dongyan Zhao. 2023. Leveraging denoised Abstract Meaning Representation for grammatical error correction. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7180–7188, Toronto, Canada. Association for Computational Linguistics.

Dhivya Chandrasekaran and Vijay Mago. 2021. Evolution of semantic similarity—a survey. *ACM Computing Surveys (CSUR)*, 54(2):1–37.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 263–270, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Susan Craw. 2017. Manhattan distance. *Encyclopedia of Machine Learning and Data Mining*, pages 790–791.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada. Association for Computational Linguistics.

Davi de Castro Reis, Paulo Braz Golgher, Altigran Soares da Silva, and Alberto H. F. Laender. 2004. Automatic web news extraction using tree edit distance. In *The Web Conference*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *International Conference on Learning Representations*.

Tao Fang, Jinpeng Hu, Derek F. Wong, Xiang Wan, Lidia S. Chao, and Tsung-Hui Chang. 2023a. Improving grammatical error correction with multimodal feature integration. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 9328–9344, Toronto, Canada. Association for Computational Linguistics.

Tao Fang, Shu Yang, Kaixin Lan, Derek F Wong, Jinpeng Hu, Lidia S Chao, and Yue Zhang. 2023b. Is chatgpt a highly fluent grammatical error correction system. *A comprehensive evaluation. ArXiv, abs/2304.01746*.

Wael H Gomaa, Aly A Fahmy, et al. 2013. A survey of text similarity approaches. *international journal of Computer Applications*, 68(13):13–18.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *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. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Yuquan Le, Zhi-Jie Wang, Zhe Quan, Jiawei He, and Bin Yao. 2018. Acv-tree: A new method for sentence similarity modeling. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4137–4143. International Joint Conferences on Artificial Intelligence Organization.

Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. 2023a. Unified demonstration retriever for in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4644–4668, Toronto, Canada. Association for Computational Linguistics.

Yinghao Li, Xuebo Liu, Shuo Wang, Peiyuan Gong, Derek F. Wong, Yang Gao, Heyan Huang, and Min Zhang. 2023b. TemplateGEC: Improving grammatical error correction with detection template. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6878–6892, Toronto, Canada. Association for Computational Linguistics.

Pengyu Liu, Tinghao Feng, and Rui Liu. 2022. Quantifying syntax similarity with a polynomial representation of dependency trees. *arXiv preprint arXiv:2211.07005*.

Mengsay Loem, Masahiro Kaneko, Sho Takase, and Naoaki Okazaki. 2023. Exploring effectiveness of GPT-3 in grammatical error correction: A study on performance and controllability in prompt-based methods. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 205–219, Toronto, Canada. Association for Computational Linguistics.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Machine Learning: ECML 2006*, pages 318–329, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. 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. Association for Computational Linguistics.

OpenAI. 2023. GPT-3.5 API. `https://platform.openai.com/docs/models/gpt-3-5`.

Masanori Oya. 2020. Syntactic similarity of the sentences in a multi-lingual parallel corpus based on the Euclidean distance of their dependency trees. In *Proceedings of the 34th Pacific Asia Conference on Language, Information and Computation*, pages 225–233, Hanoi, Vietnam. Association for Computational Linguistics.

Şaziye Betül Özateş, Arzucan Özgür, and Dragomir Radev. 2016. Sentence similarity based on dependency tree kernels for multi-document summarization. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2833–2838, Portorož, Slovenia. European Language Resources Association (ELRA).

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at trec-3. In *Text Retrieval Conference*.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. 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. Association for Computational Linguistics.

Maksym Tarnavskyi, Artem Chernodub, and Kostiantyn Omelianchuk. 2022. Ensembling and knowledge distilling of large sequence taggers for grammatical error correction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3842–3852, Dublin, Ireland. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

SVN Vishwanathan, Alexander Johannes Smola, et al. 2004. Fast kernels for string and tree matching. *Kernel methods in computational biology*, 15(113-130):1.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2023. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Xiuyu Wu and Yunfang Wu. 2022. From spelling to grammar: A new framework for Chinese grammatical error correction. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 889–902, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. 2023. Self-adaptive in-context learning: An information compression perspective for in-context example selection and ordering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1423–1436, Toronto, Canada. Association for Computational Linguistics.

Konstantin Yakovlev, Alexander Podolskiy, Andrey Bout, Sergey Nikolenko, and Irina Piontkovskaya. 2023. GEC-DePenD: Non-autoregressive grammatical error correction with decoupled permutation and decoding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1546–1558, Toronto, Canada. Association for Computational Linguistics.

Lingyu Yang, Hongjia Li, Lei Li, Chengyin Xu, Shutao Xia, and Chun Yuan. 2023. LET: Leveraging error type information for grammatical error correction. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5986–5998, Toronto, Canada. Association for Computational Linguistics.

I	II	1-shot P	1-shot R	1-shot $F_{0.5}$	2-shot P	2-shot R	2-shot $F_{0.5}$	4-shot P	4-shot R	4-shot $F_{0.5}$	8-shot P	8-shot R	8-shot $F_{0.5}$
-	Rand.	54.7	21.2	41.6	58.0	42.3	54.0	59.1	48.2	56.6	60.9	49.3	58.2
	BM25	55.7	25.2	**44.9**	57.5	42.5	53.7	59.7	47.7	56.8	60.4	47.8	57.4
	BERT	55.9	22.5	43.1	58.0	39.0	52.8	58.6	45.4	55.4	60.7	48.0	57.6
	T. K.	51.5	17.7	37.3	57.9	44.1	54.5	57.9	47.5	55.5	61.7	47.0	58.1
	Poly.	54.7	21.3	41.6	58.6	41.4	54.1	59.6	49.5	57.2	60.5	48.5	57.6
	W. Poly.	52.3	20.6	40.0	58.6	42.9	**54.6**	60.1	49.2	**57.5**	61.0	49.8	**58.4**
BM25	T. K.	57.9	27.3	**47.3**	60.5	44.7	**56.5**	62.2	45.7	58.0	62.5	45.3	58.1
	Poly.	57.2	25.1	45.5	60.5	43.5	56.2	62.1	47.7	**58.6**	61.6	46.7	57.9
	W. Poly.	57.1	24.6	45.1	60.7	43.7	56.3	61.4	47.7	58.1	62.7	47.7	**59.0**
BERT	T. K.	58.3	25.1	**46.1**	59.9	42.7	55.4	60.7	46.3	57.1	63.1	46.2	58.8
	Poly.	56.0	24.7	44.7	59.3	43.8	55.4	60.5	47.6	57.4	61.9	47.7	58.4
	W. Poly.	57.1	24.9	45.4	59.5	44.6	**55.8**	61.0	48.3	**57.9**	62.8	47.8	**59.1**

Table 9: Results of different numbers of shots on CoNLL-14 test set.

Ying Zhang, Hidetaka Kamigaito, and Manabu Okumura. 2023a. Bidirectional transformer reranker for grammatical error correction. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3801–3825, Toronto, Canada. Association for Computational Linguistics.

Yue Zhang and Zhenghua Li. 2022. Csyngec: Incorporating constituent-based syntax for grammatical error correction with a tailored gec-oriented parser. *arXiv preprint arXiv:2211.08158*.

Yue Zhang, Zhenghua Li, Zuyi Bao, Jiacheng Li, Bo Zhang, Chen Li, Fei Huang, and Min Zhang. 2022a. MuCGEC: a multi-reference multi-source evaluation dataset for Chinese grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3118–3130, Seattle, United States. Association for Computational Linguistics.

Yue Zhang, Bo Zhang, Zhenghua Li, Zuyi Bao, Chen Li, and Min Zhang. 2022b. SynGEC: Syntax-enhanced grammatical error correction with a tailored GEC-oriented parser. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2518–2531, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023b. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*.

A Results of different numbers of shots on CoNLL-14

The results of different numbers of shots on CoNLL-14 test set are shown in Table 9. For results on BEA-19 test set, please refer to Section 5.1.