GEC-Agent: Tool-Augmented Large Language Models for Grammatical Error Correction

Anonymous ACL submission

Abstract

In the era of large language models (LLMs), utilizing these models to address a variety of Natural Language Processing (NLP) tasks has emerged as a focal point of research. However, applying LLMs to the Grammatical Error Correction (GEC) task remains challenging. In this paper, we introduce GEC-Agent, a novel framework designed to effectively leverage the inferential and syntactic capabilities of LLMs while integrating external tools and rule-based approaches to enhance correction accuracy. The framework incorporates grammar and retrieval tools to identify and correct grammatical errors effectively, and implements a reflection mechanism to mitigate overcorrection. GEC-Agent dynamically selects appropriate tools to optimize the correction process and ensures consistency with the original text's style. Our experiments on the CoNLL-2014 and JLFEG datasets demonstrate that GEC-Agent outperforms the few-shot method, using the same large language model, and achieves a higher recall rate compared to existing traditional methods with supervised learning.

1 Introduction

Grammatical Error Correction (Bryant et al., 2023) is a fundamental task in Natural Language Processing that involves automatically detecting and correcting grammatical mistakes in the text. This task is crucial not only for enhancing the quality of text but also for applications like language learning and automated writing evaluation. Over the years, various models have been proposed for GEC. Junczys-Dowmunt et al. (2018) uses the Transformer model, Kaneko et al. (2020) applies BERT, and Rothe et al. (2021) leverages T5 for GEC.

Recently, the emergence of Large Language Models has catalyzed a paradigm shift in the application of NLP technologies, leading to significant advancements. Models like GPT and LLaMA have exhibited exceptional proficiency in downstream tasks, primarily due to their capacity to capture intricate syntactic, semantic, and contextual nuances. Extensive research has been conducted on the capabilities of large language models in the task of GEC. Fang et al. (2023a) and Loem et al. (2023) have examined the performance of large language models in the task of GEC, demonstrating that LLMs possess strong capabilities in capturing syntactic and semantic nuances. Furthermore, LLMs tend to achieve higher recall rates compared to traditional models. However, a persistent challenge remains in the form of overcorrection, where grammatically correct text segments are unnecessarily modified, thereby compromising the integrity of the original sentence. Table 1 provides an example of overcorrection by an LLM.

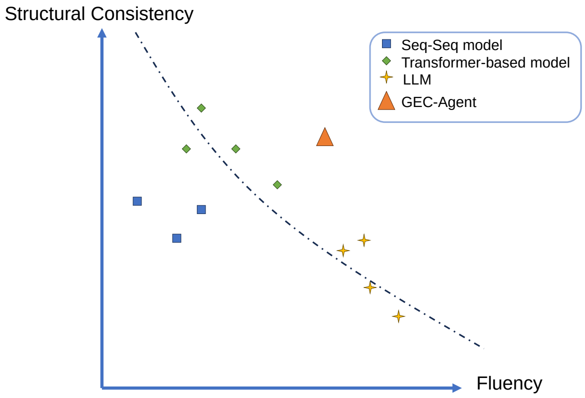


Figure 1: Traditional Seq-Seq and transformer-based models with supervised learning in GEC tasks prioritize precision, making fewer corrections to sentence structure. In contrast, large language models emphasize grammar and fluency, leading to deeper corrections but often causing over-correction. Our GEC-Agent framework attempts to accommodate both using LLM and tools.

GEC is inherently more constrained than other generative tasks due to the necessity of balancing error detection with the preservation of the original meaning and style of the sentence. As shown in

Description	Sentence
Source Sentence	The more people spend time on social media sites, the less they become ambitious.
Gold Answer	The more people spend time on social media sites, the less they become ambitious.
LLM	The more people spend time on social media sites, the less ambitious they become.

Table 1: An example demonstrating the overcorrection by large language models shows that when faced with a correct sentence, LLMs make unnecessary adjustments to the original sentence for issues like fluency or word order.

Figure 1, traditional methods with supervised learning can carefully ensure consistency in the form of input and output text but often lead to missed error corrections, whereas large models tend to ambitiously overcorrect to make sentences fluent. Simple prompting techniques fail to ensure that LLMs remain faithful to the original text, leading to a trade-off between fluency and structural fidelity.

To address these limitations, we propose GEC-Agent, a novel framework that integrates the inferential power of LLMs with rule-based and tool-assisted methods. By combining the reasoning strengths of LLMs with the precision provided by grammar rules and external tools, GEC-Agent enhances correction accuracy while preserving the original style and intent of the sentence. This hybrid approach effectively mitigates overcorrection, ensuring that the revisions are grammatically sound while maintaining stylistic consistency. The core contributions of this work are as follows:

- **LLM as a Reasoner in GEC**: For the first time in GEC, we utilize the LLM as a reasoner, responsible for generating and proposing editing operations to drive the correction process.

- **Rule/Tool-based Constraints**: We introduce rule-based and tool-based constraints to limit LLM flexibility, combining the adaptive reasoning of LLMs with the precision of strict grammatical rules.

- **Explainable and Superior Performance**: Our approach surpasses LLMs by delivering interpretable corrections, maintaining high recall, and achieving more accurate and explainable GEC outcomes.

2 Related Work

2.1 Grammatical Error Correction

Grammatical Error Correction has evolved significantly with advances in machine learning techniques.

Seq2Seq Early work primarily focuses on sequence-to-sequence models (Junczys-Dowmunt et al., 2018), which treats GEC as a translation task, translating erroneous sentences into corrected ones. Enhancements such as data synthesis and advanced reranking strategies have further improved these models (Stahlberg and Kumar, 2021a; Lichtarge et al., 2020).

Seq2Edit Seq2Edit models like GECToR (Omelianchuk et al., 2020), have since gained prominence, introducing an efficient token-level correction process that tags errors instead of rewriting entire sentences. This model reduces inference time while maintaining high accuracy, particularly in low-resource settings (Stahlberg and Kumar, 2020).

Transformer-based Transformer-based models have played a crucial role in recent developments, leveraging architectures like BART and T5 (Lewis et al., 2019; Raffel et al., 2019), which excel at handling long dependencies. These models have been fine-tuned on GEC-specific datasets, achieving state-of-the-art results. Pre-training strategies and large-scale unsupervised data have been instrumental in this improvement (Grundkiewicz et al., 2019).

Large language models LLMs such as GPT-3 and GPT-4 have been employed for GEC (Fang et al., 2023b), although they face challenges related to over-correction. Recent studies indicate that these models perform well when guided with in-context examples (Tang et al., 2024).

Syntax-aware approaches have also gained traction. SynGEC (Zhang et al., 2022b) incorporates syntactic information to guide the correction process, improving performance by exploiting sentence structures. Tang et al. (2024) uses syntactic information to select in-context examples.

Finally, data augmentation techniques have been widely adopted to address the scarcity of annotated GEC datasets. Models like that of Stahlberg and Kumar (2021b) employ synthetic data generation to create large, diverse corpora for training, which significantly boosts model performance.

2.2 Tool-Augmented LLM Agents

The development of Tool-Augmented Large Language Models (TALMs) has greatly improved LLMs' ability to perform complex tasks by leveraging external tools. Some work introduces tool integration to enhance decision-making and reasoning (Parisi et al., 2022; Schick et al., 2023; Lu et al., 2023; Mialon et al., 2023; Qin et al., 2024; Yin et al., 2024). Recent work has also focused on the iterative refinement of outputs using external tools (Madaan et al., 2023; Wu et al., 2023; Shah et al., 2022). Yao et al. (2023) emphasized the potential of combining reasoning and action capabilities in TALMs for dynamic environments. In domain-specific tasks, ChemCrow (Bran et al., 2023) and TORA (Gou et al., 2024) highlight how tool integration can enhance precision in certain fields like chemistry and mathematics.

Augmenting LLMs with domain-specific tools improves their ability to handle specialized tasks in fields. However, there have been no attempts to combine LLM and tools on GEC tasks, which could synthesize the reasoning ability of LLM with the ruled nature of tools.

3 GEC-Agent

This section outlines the design and implementation of the GEC-Agent framework, which integrates LLMs with specialized grammar tools and retrieval tools. By leveraging these components, the framework aims to improve grammatical error detection and correction while minimizing overcorrection. We will introduce GEC-Agent from four key aspects: the overall framework and logic design, the types of sentence operations, the tools integrated, and the iterative correction algorithm. Figure 2 provides an overview of the agent's operational flow.

3.1 Framework and Logic Design

We adopt LangChain (Chase, 2022) to build GEC-Agent, leveraging its modularity and seamless integration with external tools. Designed to enable LLMs to interact dynamically with external resources, LangChain provides the flexibility needed for GEC, allowing the agent to automatically select the most suitable tools based on sentence complexity. By analyzing grammatical structure and complexity, the agent invokes the appropriate tools to make precise and contextually accurate corrections.

To achieve this, we design a control logic framework that enables the agent to follow a predetermined path. Appendix B outlines the main structure of the prompt guiding the agent's operation. This prompt specifies the requirements for the GEC task, assists the agent in selecting appropriate tools based on the context, defines how the agent should perform corrections, and how it should reflect on its results after correction. Ultimately, it generates output that facilitates interaction with the LangChain framework and external tools. The control logic oversees the entire correction process, organizing it into four stages: *Thought*, *Action*, *Reflection*, and *Final Answer*. In the following paragraphs, we will introduce each of these stages in detail.

Thought In the thought stage, the agent processes the observed context and assesses whether the current correction meets the requirements. The observed context refers to the input information maintained by the LangChain framework, including initial rule constraints, each round's actions, tools' outputs, and model outputs. This information is stored as a stack of results in their generated order, without further processing. If the agent identifies the need to reflect, the agent will either move to the action stage to invoke tools or apply its own reasoning to modify the sentence. If the agent identifies the need to reflect on previous results, it will enter the reflection stage, possibly rolling back prior modifications and initiating a new round of the process.

Action In the action stage, the agent will invoke the appropriate tool and provide the input sentence to the tool. Once the tool's results are returned, the agent will observe them, and the tool's results along with the observations will be incorporated into the contextual information. After that, a new round of the process will begin.

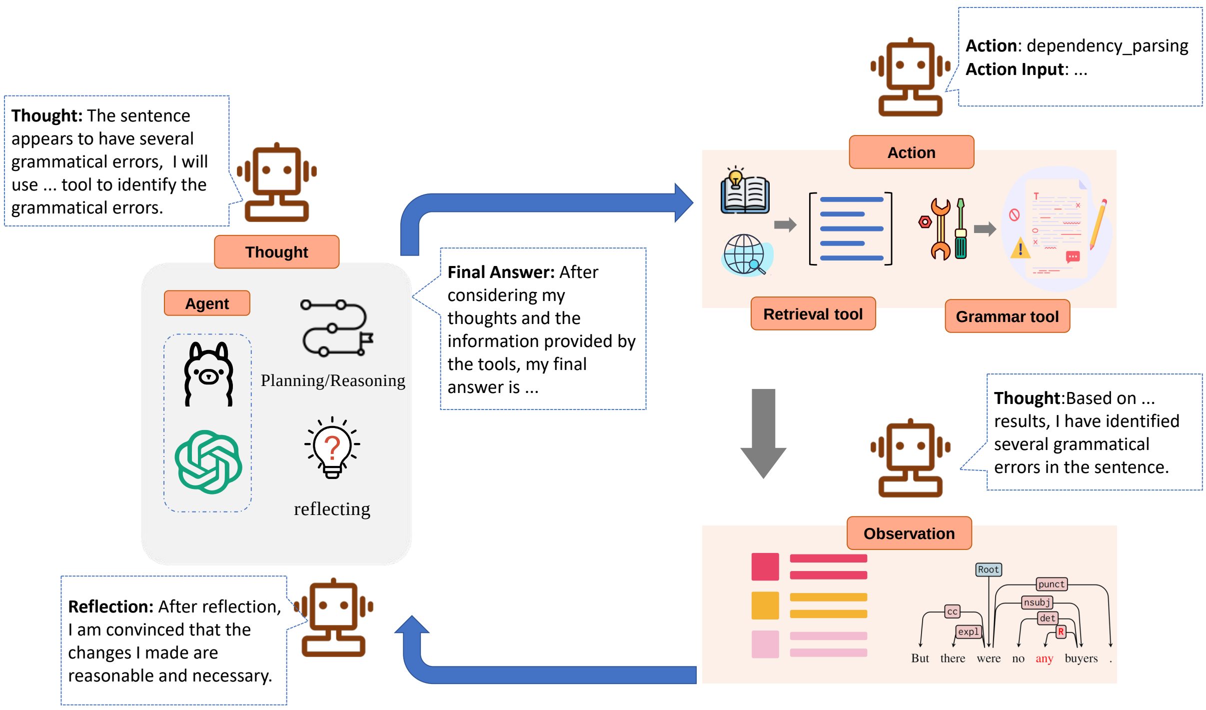


Figure 2: The GEC-Agent framework. The agent utilizes external tools to conduct deeper grammar checks or retrieve external knowledge and make corrections. By combining the inferential power of the LLM with the precision of external tools, the framework ensures accurate grammatical corrections while minimizing unnecessary changes.

Reflection Reflection is a core component of GEC-Agent, dynamically reevaluating previous corrections to determine whether they were necessary. Reflection is triggered when the agent thinks the previous changes may not have been optimal. The agent uses its internal reasoning to assess whether previous modifications were too aggressive, resulting in the loss of the original meaning or style of the sentence. If necessary, the agent will roll back certain modifications, restoring parts of the original text that were overcorrected, thus preserving the intended meaning and maintaining the accuracy and integrity of the final output.

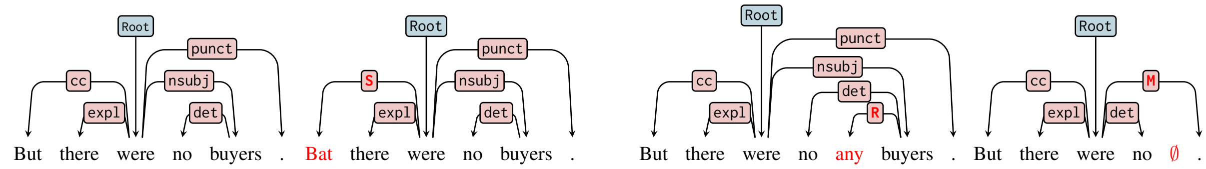
Final Answer When the large model, through its own reasoning, determines that the sentence has been correctly fixed without overcorrection, it will output the final answer.

Figure 2 illustrates the sequential relationship between the Thought, Action, Reflection, and Final Answer stages. Each stage is connected to the next through decision points based on the agent's analysis. Also, the agent decides whether to invoke an external tool, directly modify the text, or reflect on prior corrections. This control mechanism helps that corrections are both accurate and stylistically consistent with the original text, preventing overcorrection while preserving the intended meaning.

3.2 Types of Sentence Operations

In GEC, common errors can be classified into four types: *misuse*, *missing*, *redundancy*, and *word order* (Bryant et al., 2017; Zhang et al., 2022a). Grammatical error correction can be understood as a series of operations that transform an incorrect sentence into a correct one. To ensure a structured and interpretable correction process, we have limited the types of modifications that the model can make to erroneous sentences. According to Bryant et al. (2017), we define a set of core operations, each designed to handle specific types of errors:

- **Insert**: Adding missing words or phrases to the sentence.
- **Delete**: Removing redundant or incorrect words.
- **Transform**: Modifying the form of words, such as tense, singular/plural forms, or other grammatical attributes, or replacing incorrect words with appropriate ones.
- **Rearrange**: Changing the word order within the sentence.



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

Figure 3: Original illustration of GOPar from Zhang et al. (2022b). ∅ denotes the missing word.

The table below shows how these operations map to specific error types:

Error Type	Applicable Operations
Missing	Insert
Redundancy	Delete
Misuse	Delete, Transform
Word Order	Rearrange

Table 2: Mapping of GEC error types to predefined operations.

These operations form the functional backbone of the correction process, ensuring that all modifications are precise and minimize unnecessary changes. Each operation is carefully mapped to address specific error types, ensuring a targeted and efficient correction strategy. Evidently, these four types of errors can indeed be effectively resolved using the defined operations[1].

3.3 Tools

Inspired by the knowledge required by humans when correcting grammatical errors, we equipped GEC-Agent with grammar tools to provide precise grammatical knowledge and retrieval tools to supply experiential knowledge from textual data.

3.3.1 Grammar Tools Integration

To improve correction accuracy, GEC-Agent integrates two primary grammar tools: SpaCy and GOPar, each serving a distinct role in the analysis and correction of grammatical errors. These tools complement the model's capabilities, enabling a nuanced understanding of syntax and error patterns.

SpaCy SpaCy (Honnibal et al., 2020), a highly efficient NLP library, is utilized in GEC-Agent for its robust part-of-speech (POS) tagging and dependency parsing functionalities. The agent leverages SpaCy's POS tagging to identify the grammatical category of each word in a sentence, which serves as foundational information for understanding sentence structure and facilitating downstream tasks. Dependency parsing is then employed to reveal the syntactic relationships between words, enabling the agent to detect deeper grammatical issues like misaligned dependencies or incorrect phrasal structures. By integrating SpaCy's syntactic insights, GEC-Agent can accurately diagnose errors and propose corrections that adhere to grammatical rules.

GOPar GOPar (Zhang et al., 2022b) is a specialized grammatical error correction parser designed to detect and annotate substitution, redundancy, and omission errors. Unlike traditional parsers, GOPar is tailored for GEC tasks, providing a fine-grained analysis of both well-formed and erroneous sentences. In GEC-Agent, GOPar enhances the agent's ability to handle complex grammatical issues by offering detailed syntactic diagnostics, allowing the model to pinpoint the exact nature and location of errors. Through GOPar, GEC-Agent can perform sentence-level corrections while aiming to preserve the intended meaning, providing corrections that are both syntactically accurate and contextually relevant. Figure 3 illustrates three sample parses of the tool.

By integrating the syntactic information provided by SpaCy and GOPar, GEC-Agent can perform more comprehensive grammatical corrections. This tool integration enables the agent to flexibly adapt its correction strategy based on the complexity and structure of the input text, ensuring reliable grammatical error correction.

3.3.2 Retrieval Tools Integration

We also incorporate retrieval tools through the LangChain framework, leveraging DuckDuckGo[2] APIs for real-time access to external grammatical resources. Additionally, a local error sentence database built from the W&I+LOCNESS (Bryant et al., 2019) datasets allows the model to retrieve

[1]These sequential operations and the results of sequential modifications are generated by the agent through reasoning in the Thought stage, while the Action stage involves tool invocation. Please avoid conflating the two.

[2]https://duckduckgo.com

Algorithm 1 Interactive Grammatical Correction Algorithm

procedure CORRECTGRAMMAR(S(Set of Sentences), T(Set of Tools), A(Set of Actions), H(Context))
  for each $s_i \in S$ **do**
    $H \leftarrow H \cup \{\text{ExtractContext}(s_i)\}$
    while not **TerminationCondition**(H) **do**
      $a_i \leftarrow \text{DecideAction}(H, A)$ ▷ Decide to 'Think', 'Retrieve' or use a tool
      if $a_i =$ tool action **then**
        $t_i \leftarrow \text{SelectTool}(T)$
        $h_i \leftarrow \text{ApplyTool}(t_i, s_i)$ ▷ Apply selected tool to the sentence
        $H \leftarrow H \cup \{\text{ExtractContext}(h_i)\}$ ▷ Update context with the tool's result
      else
        $h_i \leftarrow \text{Think}(s_i, H)$ ▷ Internal thinking/retrieving process. The Reflection stage can be integrated into the Thought stage during implementation.
        $H \leftarrow H \cup \{\text{ExtractContext}(h_i)\}$ ▷ Update context with the result of thinking
      end if
      $s_i \leftarrow \text{modifications}(H, s_i)$ ▷ Correct the sentence according to the contextual information
    end while
  end for
  return $FinalAnswer(H)$ ▷ Return the final corrected sentences
end procedure

grammar-related examples to guide its correction decisions. To enhance the retrieval of grammar-related examples, we utilize LLaMA3.1-70B to summarize modification suggestions and the relevant grammatical knowledge for sentence pairs in the database. Through this, we can retrieve grammatical knowledge and analogous corrections through semantic similarity, by providing an erroneous sentence and the required grammatical concept. The generated data segments and the prompts provided to LLaMA3.1-70B are detailed in Appendix D. When the agent requires examples or suggestions for specific grammatical knowledge, it queries the database to retrieve grammatically or semantically similar sentences, or those with identical errors, aiding its correction decisions in complex or ambiguous scenarios. By querying external sources and the local error database, enriched with common grammatical mistakes, the model avoids unnecessary corrections, maintains precision in challenging cases, and quickly accesses past error patterns for more accurate and contextually informed corrections.

3.4 Iterative Correction Algorithm

GEC-Agent utilizes an iterative correction algorithm that progressively refines the sentence with each correction cycle. If unresolved errors or new errors from previous modifications are detected, the agent initiates another correction or reflection. This process continues until the sentence achieves an optimal state of grammatical correctness, determined dynamically by the model. The termination condition is designed to avoid unnecessary adjustments, ensuring an efficient and effective correction. For detailed algorithmic steps, refer to Algorithm 1.

4 Experiment

To rigorously assess the performance of our proposed GEC-Agent framework, we conduct comprehensive experiments across multiple benchmarks and evaluate various aspects of the model's abilities, including grammatical correction, reasoning capacity, and reflection effectiveness. We select two major GEC datasets, CoNLL-2014 (Ng et al., 2014) and JFLEG (Napoles et al., 2017), for testing, as these datasets are widely used in the GEC field and encompass a broad spectrum of linguistic complexity and error types. Moreover, the evaluation metrics of CoNLL-2014 focus more on structural consistency, while the evaluation metrics of JLFEG emphasize semantic consistency. By assessing both aspects, we can better demonstrate the capabilities of our Agent in terms of both semantics and form. We also perform an ablation study to examine the contribution of different components of our model. For the evaluation experiments, we use GPT-4o and LLaMA 3.1-70B to conduct tests on the CoNLL-

2014 and JFLEG datasets, respectively.

For the ablation experiments and tool usage analysis, we conduct tests on the CoNLL-2014 dataset using the LLaMA 3.1-70B model.

Dataset	#Sentences	%Error	Usage
W&I+LOCNESS	34,308	66	retrieval
CoNLL-14-Test	1,312	72	Testing
JFLEG-Test	747	-	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.

The proposed method is implemented using the following LLMs:

- **LLaMA 3.1** LLaMA 3.1-70B is a commonly used model of the LLaMA family, specifically designed to handle complex natural language processing tasks in multi-task scenarios.
- **GPT-4o** GPT-4o is a more efficient architecture, focusing on enhancing reasoning ability, reducing inference time, and improving context retention.

The relevant parameter settings for the large models are presented in Appendix C.

4.1 Evaluation Metrics

In order to comprehensively evaluate the performance of the GEC model, we evaluate the performance on the CoNLL-14 test set (Ng et al., 2014) using the M^2 Scorer (Dahlmeier and Ng, 2012), and evaluate the performance on the JFLEG test set using *GLEU*(Napoles et al., 2015).

4.2 Main Results

The proposed GEC-Agent framework demonstrates superior performance in the task of GEC, particularly by addressing the pervasive issue of overcorrection found in Large Language Models. By combining the inferential strengths of LLMs with the precision of rule-based and tool-augmented correction mechanisms, our approach significantly enhances correction accuracy while reducing unnecessary alterations to the original text. The experimental results across multiple benchmark datasets validate this improvement.

On the CoNLL-2014 dataset, GEC-Agent achieves an F0.5 score of 63.2, outperforming recent three-shot LLMs, and maintaining a high recall rate. The model's ability to dynamically adjust its correction strategy by integrating external grammatical tools and a reflection mechanism proves crucial in dealing with complex grammatical structures. On the JFLEG dataset, GPT-4o+GEC-Agent achieves a GLEU score of 63.4. Although it does not surpass the results of the three-shot GPT-4o on the JFLEG dataset, it still outperforms the previous traditional models, reflecting its capacity to maintain the original meaning and style of sentences while minimizing unnecessary corrections.

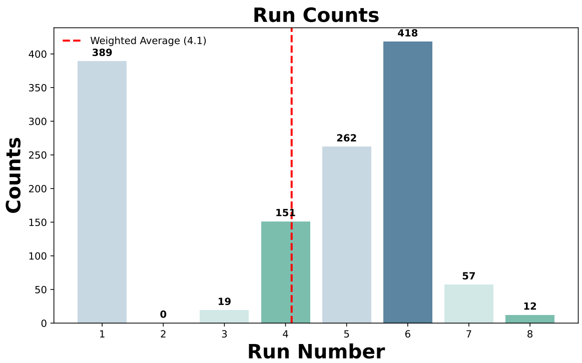


Figure 4: Distribution of reasoning iterations required to reach the final answer across the Conll-2014 dataset

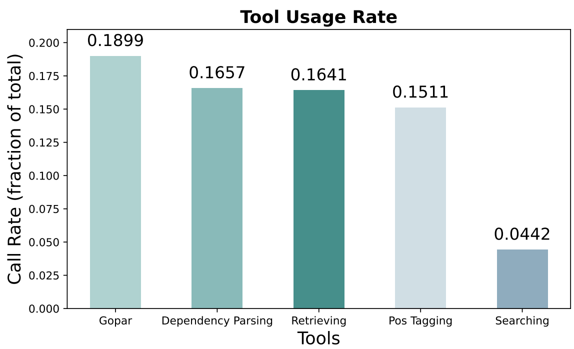


Figure 5: Tool Usage Rate

Figure 4 shows the distribution of reasoning iterations required to reach the final answer across the CoNLL-2014 dataset. From this figure, we can observe that the average reasoning path length is 4.1, with a higher number of sentences requiring only one iteration. Many sentences can arrive at the correct answer after a single reasoning step. The number of sentences requiring two iterations is zero, and one possible reason for this is that a full tool invocation step may exceed two iterations. Figure 5 displays the *Tool Usage Rate* of various tools during Agent execution. The GOPar tool, which is most related to grammatical errors, has the highest number of invocations, while the search

System	CoNLL-14			JFLEG
	P	R	$F_{0.5}$	GLEU
Transformer (Fang et al., 2023b)	60.1	36.6	53.3	55.4
T5 large (Rothe et al., 2021)	72.2	51.4	66.8	62.8
GECToR (Omelianchuk et al., 2020)	75.6	44.5	66.3	58.6
ChatGPT zero-shot (Fang et al., 2023b)	48.5	58.9	50.3	-
ChatGPT zero-shot CoT (Fang et al., 2023b)	50.2	59.0	51.7	61.4
LLaMA-3.1-70B three-shots	55.1	58.7	55.8	62.1
LLaMA-3.1-70B +GEC-Agent	60.0	48.4	57.3	62.7
GPT-4o three-shots	59.0	55.4	58.2	64.1
GPT-4o +GEC-Agent	67.6	50.3	63.2	63.4

Table 4: Results of state-of-the-art GEC systems and our proposed methods on two datasets: CoNLL-14 (evaluated using Precision (P), Recall (R), and $F_{0.5}$) and JFLEG (evaluated using GLEU).

tool is invoked less frequently.

Condition	P	R	$F_{0.5}$
Remove Grammar Tools	58.7	43.8	55.0
Remove Retrieval Tools	57.1	47.9	55.0
Remove both	53.6	46.4	52.0
Keep all	**60.0**	**48.4**	**57.3**

Table 5: Ablation Study Results

4.3 Ablation Study

The ablation study further underscores the importance of tool integration within GEC-Agent. When either grammatical tools or retrieval mechanisms are removed, there is a significant drop in performance, particularly in precision. The $F_{0.5}$ score drops from 57.3 to 52.0 when both components are excluded, highlighting the indispensable role of external tools in ensuring correction accuracy. Retaining all components allows the model to adapt its correction strategy dynamically, providing robust performance across a broader range of grammatical errors.

4.4 Case Study

We demonstrate two types of case studies: tool-assisted correction and reflection. They are shown in Appendix A. In tool-assisted correction, the large model uses external tools to detect and fix grammatical errors with higher precision. In Example A.1, the large model invokes the GOPar tool, which returns a syntax tree annotated with grammatical error information. The model observed these grammatical errors and reasoned accordingly. For different types of errors, the model applied predefined operation types to modify the sentence.

In reflection, the model reassesses prior corrections, retracting unnecessary changes to maintain the original meaning and style. In Example A.4, the model evaluates each previous modification, and when it detects that "requires" was an overcorrection of the original text, the model identifies this and reverts the modification.

These examples also demonstrate that our method offers excellent interpretability, making it easier for non-native speakers to receive correct and comprehensible error corrections, which facilitates both comprehension and learning when encountering grammatical mistakes.

5 Conclusion

In this work, we propose a novel approach to Grammatical Error Correction through the integration of large language models with external grammar tools and a reflection mechanism, resulting in the creation of the GEC-Agent. The results in our experiments demonstrate the significant advantages of GEC-Agent: by combining the reasoning power of LLMs with the precision of external grammatical tools and the adaptability of the reflection mechanism, GEC-Agent gets an effective grammatical correction while minimizing overcorrection, preserving the original semantic and stylistic integrity of the text, and showcasing the potential of tool-augmented large model frameworks in GEC tasks.

6 Limitations

Despite promising results, the GEC-Agent system has several limitations. The reliance on external grammar tools and retrieval mechanisms poses efficiency challenges, particularly in large-scale or real-time scenarios. Additionally, the evaluation of publicly available datasets like CoNLL-14 and JFLEG may not fully capture the range of real-world grammar errors, highlighting the need for testing on more diverse and domain-specific datasets. Furthermore, we acknowledge the language limitations of our current system. Due to the availability and robustness of current tools, GEC-Agent currently supports English. We are working on extending basic GEC capabilities to low-resource languages through rule-based grammar guidance. Lastly, while the GEC-Agent reduces overcorrection, it does not fully eliminate the problem. There are still cases where the model modifies correct sentences unnecessarily, especially in complex syntactic structures or with rare grammatical constructions. More experiments are needed to improve the performance.

References

Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2023. Chemcrow: Augmenting large-language models with chemistry tools. *Preprint*, arXiv:2304.05376.

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. 2023. Grammatical error correction: A survey of the state of the art. *Computational Linguistics*, pages 643–701.

Harrison Chase. 2022. LangChain.

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.

Tao Fang, Shu Yang, Kaixin Lan, Derek F. Wong, Jinpeng Hu, Lidia S. Chao, and Yue Zhang. 2023a. Is chatgpt a highly fluent grammatical error correction system? a comprehensive evaluation. *Preprint*, arXiv:2304.01746.

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 preprint arXiv:2304.01746*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2024. Tora: A tool-integrated reasoning agent for mathematical problem solving. *Preprint*, arXiv:2309.17452.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

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.

Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2020. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4248–4254, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Preprint*, arXiv:1910.13461.

Jared Lichtarge, Chris Alberti, and Shankar Kumar. 2020. Data weighted training strategies for grammatical error correction. *Preprint*, arXiv:2008.02976.

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.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. *Preprint*, arXiv:2304.09842.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *Preprint*, arXiv:2303.17651.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented language models: a survey. *Preprint*, arXiv:2302.07842.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593, Beijing, China. Association for Computational Linguistics.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 229–234, Valencia, Spain. Association for Computational Linguistics.

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.

Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. Talm: Tool augmented language models. *Preprint*, arXiv:2205.12255.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. 2024. Tool learning with foundation models. *Preprint*, arXiv:2304.08354.

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

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.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Preprint*, arXiv:2302.04761.

Dhruv Shah, Blazej Osinski, Brian Ichter, and Sergey Levine. 2022. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. *Preprint*, arXiv:2207.04429.

Felix Stahlberg and Shankar Kumar. 2020. Seq2Edits: Sequence transduction using span-level edit operations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5147–5159, Online. Association for Computational Linguistics.

Felix Stahlberg and Shankar Kumar. 2021a. Synthetic data generation for grammatical error correction with tagged corruption models. *Preprint*, arXiv:2105.13318.

Felix Stahlberg and Shankar Kumar. 2021b. Synthetic data generation for grammatical error correction with tagged corruption models. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 37–47, Online. Association for Computational Linguistics.

Chenming Tang, Fanyi Qu, and Yunfang Wu. 2024. Ungrammatical-syntax-based in-context example selection for grammatical error correction. *Preprint*, arXiv:2403.19283.

Yue Wu, So Yeon Min, Yonatan Bisk, Ruslan Salakhutdinov, Amos Azaria, Yuanzhi Li, Tom Mitchell, and Shrimai Prabhumoye. 2023. Plan, eliminate, and track – language models are good teachers for embodied agents. *Preprint*, arXiv:2305.02412.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Xunjian Yin, Xinyi Wang, Liangming Pan, Xiaojun Wan, and William Yang Wang. 2024. Gödel agent: A self-referential agent framework for recursive self-improvement. *Preprint*, arXiv:2410.04444.

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.

A Some detailed examples

A.1

Action: GOPar_dependency_parsing
Action Input: "Therefore, the social media is not safe which is bad to teenager 's future development."

Observation: Based on the GOPar dependency parsing results, I notice that the sentence has several errors marked as R (Redundant), M (Missing), and S (Substituted). Specifically, the article "the" is marked as redundant, the word "which" is marked as missing, and the preposition "to" and the possessive form "'s" are marked as substituted.
Thought: Based on the observation, I will correct the sentence by addressing the identified issues. First, I will remove the redundant article "the" before "social media." Next, I will add a comma before "which" to properly separate the clauses. I will replace the preposition "to" with "for," as it is the appropriate preposition in this context. Finally, I will correct the possessive form by changing "'s" to "s'."
Final Answer: "Therefore, social media is not safe, which is bad for teenagers' future development."

A.2

Action: duckduckgo_search
Action Input: "typical phrase with 'talk'"

Observation: After searching, I found that the fixed combination referred to in the sentence is "talking to people." Therefore, I can rewrite the sentence as: "One can talk to some people some of the time, but not all people all of the time."

Final Answer: "One can talk to some people some of the time, but not all people all of the time."

A.3

Action: pos_tagging
Action Input:"My advice to any one start to learn this sport to be careful of the accident ."

Observation: "start" is VERB.
Thought: Need to change ’start’ to ’starting’

A.4

Reflection: Based on the previous correction result, I will change "requires" back to "needs" since "requires" was an overcorrection and "needs" maintains the original tone. Additionally, I will retain the other corrections, such as combining "any one" into "anyone," changing "start learn" to "starting learning," and modifying "carefully from the accident" to "be careful of the accident," as these were valid improvements.

Final Answer: "My advice to anyone starting learning this sport is to be careful of the accident because it needs more concentration from the person to ride."

B Main Prompt for the GEC-Agent

```
PREFIX = """
You are an expert linguist specialized in identifying and correcting grammatical and spelling errors. Your task is to find any grammatical or spelling errors in the given sentence and correct them using the provided tools, applying the minimal necessary edits. **Do not make any additional changes** such as adding content, replacing synonyms, rephrasing sentences for better flow, or altering the original meaning.
"""

FORMAT_INSTRUCTIONS = """
You must respond using one of the following formats:

1. "Thought, Action, Action Input" format:
   - Thought: Reflect on your progress and decide the next action.
   - Action: Specify the tool to use, selecting from [{tool_names}].
   - Action Input: Provide the input for the chosen tool.

OR

2. "Final Answer" format:
   - Final Answer: Provide the corrected sentence without grammatical or spelling errors.

**Only a single complete format should be used in each response.**
"""

QUESTION_PROMPT = """
Identify any grammatical or spelling errors in the sentence and correct them using the following tools:

{tool_strings}

Use the most appropriate tool available for each correction.

**IMPORTANT:** Follow these steps in order and strictly adhere to the guidelines to ensure minimal modifications:

1. **Grammar and Spelling Check:** Examine the sentence for the following issues:
   - Excessive or incorrect use of prepositions or articles
   - Missing prepositions, articles, or verbs
   - Tense and voice inconsistencies
   - Capitalization errors
   - Spelling mistakes
   - Missing or incorrect punctuation
   - Singular and Plural Errors: Incorrect usage of singular or plural forms.
```

```
    - Possessive Case Errors: Incorrect usage of possessive forms.
    - Subject-Verb Agreement Errors: Ensure that the subject and verb agree in number and person.
    - Sentence Structure Errors:
      - Sentence Fragments: Incomplete sentences lacking main components.
      - Run-on Sentences: Improperly connected independent clauses.
    - Pronoun-Antecedent Agreement Errors: Ensure pronouns agree with their antecedents in number and gender.
    - Incorrect Use of Conjunctions: Proper usage of coordinating and subordinating conjunctions.
    - Misuse of Adjectives and Adverbs: Correct application of adjectives and adverbs to modify appropriate words.
    - Redundancy and Repetition: Eliminate unnecessary repetition of words or phrases.
    - Improper Negation: Avoid double negatives and ensure clear negation structures.

    *Note:* Do not consider word order or synonym issues as grammatical errors.

2. **No Errors Found:** If no grammatical or spelling errors are detected, return the original sentence.

3. **Minimal Modification:** Make **only one modification at a time**, applying the least intrusive change necessary to correct the error.

4. **Avoid Unnecessary Changes:** **Do not make any modifications** that do not address a grammatical or spelling error. **Do not add, remove, or replace words** beyond what is necessary for correction.

5. **Validation:** After each modification, **reflect to ensure it meets the above requirements**. If it does not, withdraw the modification and do not apply it.

6. **Detailed Reflection:** At the end of each step, provide a **detailed reflection** assessing whether the current action complies with the requirements. **Explain your evaluation clearly**, ensuring that no overediting has occurred.

**Do not skip any of these steps. Do not deviate from the instructions. Do not provide additional explanations, examples, or alternative formats. Do not simulate tool outputs or engage in reasoning loops.**

Sentence: {input}
"""

SUFFIX = """
Thought: {agent_scratchpad}
"""

FINAL_ANSWER_ACTION = "Final Answer:"
```

Listing 1: Main Prompt for the GEC-Agent

This prompt specifies the requirements for the GEC task, defines how the agent should perform corrections, and how it should reflect on its results after correction.

C Model parameter settings

Parameter	Value
Temperature	0.0
Top-p	0.3
Max Tokens	1024

Table 6: Parameter Settings for LLMs

For tasks like grammatical error correction, precision and consistency are paramount. Throughout this paper, the temperature parameter for LLMs is consistently set to 0.

D Retrieval Prompts and Data Segments

```
"""
# Task Description:
You are an English grammar expert. Analyze sentence pairs containing an **erroneous sentence** and its **corrected version**, and extract:
1. **Grammar Knowledge**: Rules or error types (e.g., subject-verb agreement, missing article).
2. **Modification Type**:
    - Insert: Adding missing words or phrases.
    - Delete: Removing redundant or incorrect words.
    - Transform: Modifying or replacing incorrect words.
    - Rearrange: Adjusting word order for correctness.
3. **Structured Examples**:
    - Sentence Pair: Erroneous sentence -> Corrected sentence.
    - Word Pair: Erroneous word -> Corrected word.
    - Abstract Pattern: Generalized form for reuse.

---

## Example Output:
### Example 1
- **Grammar Knowledge**: Subject-Verb Agreement
```

Table 7: Grammar Knowledge and Examples for Database Retrieval

Grammar Knowledge	Modification Type	Sentence Pair	Word Pair
Missing Article	Insert	**Incorrect:** He bought apple. **Correct:** He bought an apple.	[None] → an
Subject-Verb Agreement	Transform	**Incorrect:** Public transport provide... **Correct:** Public transport provides...	provide → provides
Capitalization	Transform	**Incorrect:** i am john from canada. **Correct:** I am John from Canada.	i → I
Adverb Placement	Rearrange	**Incorrect:** I like very much this sport. **Correct:** I like this sport very much.	very much → placed after like
Verb Tense Consistency	Transform	**Incorrect:** It must be play. **Correct:** It must be played.	play → played
Preposition Usage	Transform	**Incorrect:** She gave the book for him. **Correct:** She gave the book to him.	for → to

```
- **Modification Type**: Transform
- **Sentence Pair**: "She go to school."
    -> "She goes to school."
- **Word Pair**: go -> goes

### Example 2
- **Grammar Knowledge**: Missing Article
- **Modification Type**: Insert
- **Sentence Pair**: "He bought apple."
   -> "He bought an apple."
- **Word Pair**: [None] -> an
"""
```

Listing 2: Prompt for Retrieval-friendly Grammar Database

This prompt instructs the large model to summarize the grammatical knowledge involved in the sentence pair modifications within the dataset, facilitating its use for retrieval.

Table 7 shows the grammatical knowledge and related examples used for database retrieval. The table includes various types of grammatical errors, correction methods, sentence pairs illustrating incorrect and corrected forms, as well as the corresponding word-level modifications. These examples provide a structured and clear reference, enabling the system to retrieve relevant corrections and apply appropriate fixes based on similar patterns in the input text.