Dr. B. C. Roy Polytechnic

BCRP Journal of Innovative Research in Science and Technology (BJIRST)

A peer-reviewed open-access journal
ISSN: 2583-4290
Journal homepage: https://bcrpjournal.org/

BJIRST

Q-learning-Driven Policy Optimization for Grammar Correction utilizing Transformer-Based Language Models

Rima Dutta
Dept. of Computer Science and Technology
Dr. B.C. Roy Polytechnic
Durgapur, India
rima.dutta@bcrec.ac.in

Santu Kundu
Dept. of Computer Science and Technology
Dr. B.C. Roy Polytechnic
Durgapur, India
santu.kundu@bcrec.ac.in

Sarada Mallik
Dept.of Computer Science and Technology
Dr. B.C. Roy Polytechnic
Durgapur, India
sarada.mallik@bcrec.ac.in

Abhishek Pal
Dept. of Computer Science and Technology
Dr. B.C. Roy Polytechnic
Durgapur, India
abhishek.pal@bcrec.ac.in

Saikat Chatterjee
Dept. of Computer Science and Technology
Dr. B.C. Roy Polytechnic
Durgapur, India
saikat.chatterjee@bcrec.ac.in

Arnob Dutta
Dept. of Computer Science and Technology
Dr. B.C. Roy Polytechnic
Durgapur, India
arnobdutta0000@gmail.com

ABSTRACT

This paper proposes a grammatical error correction (GEC) framework that combines effectively the strengths of reinforcement learning with those of transform-based language models. The uniqueness of this research effort specifically resides in using a policy-level Q-learning mechanism to adaptively re-rank potential corrections from a generative model, going beyond traditional approaches that rely solely on T5 for error corrections in text, for example, or even rely on a mere *re-rank* with a model like GPT2. This proposed scheme uses a fine-tuned version of T5 for error corrections as a generative model that can provide a list of potential corrections, and a separate GPT2 model that will provide an implicit reward as a judge of grammatical fluency. Subsequently, an agent algorithm will help develop an optimal policy that fills up a Q-table that associates error states with superior actions for error corrections. To serve as a preliminary indication of feasibility, this research effort proposes a qualitative analysis with a targeted data set due to inherent limitations in scope.

Keywords— Grammatical Error Correction (GEC), Q-learning, T5 (Text-To-Text Transfer Transformer), Reinforcement Learning (RL), Natural Language Processing (NLP)

1. INTRODUCTION

It is very essential to communicate effectively in written form. Grammatical mistakes often affect clarity and professionalism in many areas. Grammatical Error Correction (GEC) systems automatically identify and correct grammatical errors. Such systems are important in a number of applications ranging from educational applications, writing assistants, and post-editing machine translation. Traditionally, GEC approaches have evolved from using rule-based parsers that relied on linguistic rules introduced by experts, to more flexible statistical methods that employed n-gram language models and phrase-based translation. While useful, prior systems struggled to capture the complexities of natural language, particularly with error variations and fluency in corrections. The rise of deep learning to Natural Language Processing (NLP) prompted a new generation of GEC systems based on neural GEC models, particularly sequence-to-sequence (Seq2Seq) models [1]. These models, particularly those employing attention mechanisms, represent a significant improvement in performance over statistical methods and learn complex mappings between incorrect and correct sentences from large corpora.

Since the introduction of large language models (LLMs) based on the Transformer architecture over recent years, they have had an explosion in popularity and have revolutionized natural language processing (NLP). Models like OpenAI's GPT-2 and Google's T5 have shown phenomenal performance in understanding, generating, and editing text, pushing the limits of what is possible in fields such as question-answering, translation, and summarization. On the other hand, reinforcement learning (RL) techniques -- in particular Q-leaning have shown remarkable ability to improve decision making at unpredictable situations. These techniques are used in areas such as natural language generation and dialogue systems where feedback-based learning and sequential decisions are important.

In this paper, we propose a novel Q-learning-based policy optimization framework that exploits the

complementary generative and evaluative capabilities of transformer-based language models (LMs) toward more effective GEC [2]. In our work, we tackle the nontrivial problem of choosing the most plausible correction in a vast space of candidates produced by state-of-the-art neural models. We advocate a hybrid approach where fine-tuned T5 models work as a corrector, generating various grammatical correction candidates and GPT-2 model operates as a judge that offers the real-time grammaticality scores according to the language model likelihood. This feedback is then employed by a Q-learning agent to learn an optimal policy on the fly, such that the best correction for a given erroneous input can be efficiently identified and selected through scoring actions in a Q-table.

The novelty at its core is using Q-learning to explore the action space of possible corrections, learning dynamically which corrective actions have the highest grammatical quality from the judgment of a highly developed language model. The main result of this work is the creation of an adaptive GEC system that exploits learned policies to drive efficient inference, providing an avenue towards more context-dependent and robust error correction, especially valuable for coping with repeated error patterns and enhancing overall correction quality as opposed to solely generative or rule-based approaches. The next few sections of this paper present the related work, our system's specific architecture and methodology, its mathematical representation, experimental procedure, and a detailed discussion of our results and future research directions.

2. RELATED WORK

The history of the field of Grammatical Error Correction (GEC) is rich, developing through a number of clear-cut methodological paradigms before the recent deep learning period. Appreciation of these earlier and concurrent work is necessary in order to place our suggested Q-learning framework appropriately.

2.1. Rule-Based and Statistical GEC Approaches

Early GEC systems were mostly dependent upon hand-coded linguistic rules and lexicons. Such rule-based ones entailed explicit grammar rule programming with large amounts of expert knowledge and maintenance [3]. Although accurate for the known error patterns, they fared poorly with ambiguity, unknown errors, and the large complexities of natural language. Subsequently, statistical GEC approaches became prominent. These tended to pose GEC as a machine translation problem, rewriting an "incorrect" sentence to produce a "correct" one. Methods such as n-gram language models, phrase-based translation, and error classification using statistical classifiers (such as Support Vector Machines) were used. Tools such as Flesch-Kincaid readability tests or spell checkers recognizing frequent spelling mistakes belong to this group.

2.2. Neural Network-Based GEC (Seq2Seq, Transformers)

The revolution of deep learning actually revolutionized GEC. RNNs, especially LSTMs and GRUs, within sequence-to-sequence (Seq2Seq) models became the norm. These models were able to train to transform an input sequence (erroneous sentence) to an output sequence (correct sentence) end-to-end. The addition of attention mechanisms further improved these models by enabling them to attend specifically to important regions of the input while generating the output, greatly improving performance as well as being able to work with longer sentences [4]. Transformer-based GEC models are currently the state-of-the-art, with high scores on benchmarking datasets by learning intricate grammatical transformations effectively. These models tend to work by fine-tuning pre-trained language models on GEC-specific data.

2.3. Applications of Large Language Models in GEC

Large Language Models like T5 and GPT-2 have shown impressive performance on a various NLP tasks. Their large size, strong pre-training on huge text datasets, and Transformer architecture allow them to produce strongly coherent and well-formed text. LLMs are applied in various manners in GEC. Encoder-decoder models fine-tuned like T5 can actually carry out GEC as a text-to-text task to produce corrections. Generative LLMs, such as GPT-2, can be trained to judge grammatical correctness in terms of sentence likelihood or by ranking candidate outputs generated by perplexity or log-probability. These models are not only great correctors but also advanced judges of linguistic quality, capable of picking up subtle grammatical ins and outs that better language models may overlook.

2.4. Reinforcement Learning in Natural Language Processing (Beyond GEC)

Reinforcement Learning (RL) has found acceptance in NLP for the purpose of sequential decision-making and optimization by interacting with an environment. Less prevalent in direct GEC usage, RL has been used effectively in a range of domains, including dialogue management, where agents learn the best conversational policies as per user feedback. It has also been applied in text generation, where a reward model rates how good generated text is, and the generator is trained with methods such as policy gradient methods [5]. For example, RL has been utilized to make generated summaries or translations more aligned with human preference or certain criteria, learning to generate outputs that have maximum specified reward.

2.5. Hybrid and Adaptive Correction Systems

The use of several techniques, or hybrid systems, has frequently provided effective GEC solutions. Such systems combine the advantages of various approaches, for example, rule-based components for frequent errors and neural networks for harder ones. Adaptive correction systems, which are a subgroup of hybrid methods, emphasize learning and self-improvement over time, often tailored to an individual user or particular error patterns. Our research belongs to this type by clearly presenting a Q-learning agent to learn best correction policies adaptively. Where earlier hybrid systems may average varying correction modules, our innovation is applying an RL agent to learn to optimize combining or choosing among the outputs of powerful LLMs, rendering the system adaptive to certain states (erroneous sentences) and their associated optimal actions (corrections) over time based on explicit reward signals. This is a step beyond straightforward ensemble methods to a feedback-informed, research directions.

Although these above works lay down the foundation for hybrid and adaptive approaches, a certain void in methodology still needs to be filled with regards to applying reinforcement learning as an adaptive reranking policy in GEC. Although previous approaches for these systems were not aimed at training an agent to find out which of their high-probability candidate was best for a judge model of similar capabilities, our effort fills this void by showing that a tabular Q-learning algorithm can solve this problem of arbitration of high-quality neural output models.

3. SYSTEM ARCHITECTURE AND METHODOLOGY

This section describes the design and working principles of our Q-learning-based grammar correction system, which we call "Q-grammar." We outline the constituent modules and their synergistic integration in a reinforcement learning framework.

3.1. Overview of the Q-grammar Framework

The Q-grammar framework is constructed as a hybrid system, combining the text generation and evaluation strengths of large language models with the adaptive learning capabilities of Q-learning for best correction selection. Its overall goal is to accept an error-containing input sentence and generate its most grammatically correct and fluent equivalent. The intelligence of the system lies in how it learns from repeated correction efforts and can rectify errors more effectively and efficiently as time goes on.

3.1.1. States, Actions, and Rewards in GEC Context

In our reinforcement learning framework, the fundamental elements are given by:

States (S): Every different incorrect sentence that the system comes across represents a different state. The actions of this agent are a result of understanding the current grammatical error, which then becomes a basis of decisions. In our proof-of-concept experiment, in order to clearly identify common recurring mistakes corresponding to their optimal corrections, we consider using the whole string of a sentence as a state. This, as acknowledged, has limitations in terms of being able to generalize to other sentences, which becomes a challenge that will be squarely taken up in our future work.

Actions (A): An action is to choose a particular correction candidate created for the given wrong sentence. The aim of the agent is to make the choice which results in the most correct output from our system.

Rewards (R): The reward signal signifies the quality of the responses. In our model, we have calculated this using the grammatical fluency score. This score is given by the judge module. This identifies the extent to which a candidate sentence follows grammatical rules and natural language patterns.

3.2. The Corrector Module

The Corrector module has the job of suggesting a varied set of possible corrections for a given input sentence. It is the agent's chief provider of possible "actions" for the Q-learning agent.

3.2.1. Role and Functionality

The primary role of the Corrector is to convert a flawed input sentence into a set of several grammatically enhanced alternatives. It does not determine which of these corrections is optimal; it merely produces a set of options from which the Ranking Agent picks. This allows the agent to have a rich source of alternatives, making it more probable that a high-quality correction will be found.

3.2.2. Implementation Details: Happy Transformer and T5 Model

Our Corrector module is applied with the HappyTextToText interface from the happytransformer library. This library includes a simple wrapper for fine-tuned Hugging Face Transformer models. We specifically employ the vennify/t5-base-grammar-correction model, which is T5 (Text-To-Text Transfer Transformer) base model fine-tuned for the GEC task. T5's encoder-decoder architecture makes it very well-suited for text-to-text transformation tasks such as grammar correction.

3.2.3. Candidate Generation Strategy (Sampling with do_sample=True)

To produce a huge range of num_candidates corrections, the Corrector uses a sampling-based approach. Rather than using num_return_sequences, which in some cases gives repeated outputs, the model is invoked multiple times. For every invocation, TTSettings are set with do_sample=True, accompanied by num_beams=5, top_k=50, and top_p=0.95. These specifications make the T5 model follow different generation routes, resulting in diverse correction candidates even from the same input sentence. The max_length parameter is dynamically set so that generated candidates are well-sized compared to the input sentence.

3.3. The Judge Module

The Judge module serves as the arbiter of grammatical quality, providing the crucial feedback mechanism for the Q-learning agent.

3.3.1. Role and Functionality (Grammaticality Scoring)

The Judge's primary function is to give a quantitative score to a sentence based on its grammaticality and fluency. The Judge's score is most crucial because it directly corresponds to the reward signal to be used in reinforcement learning. A lower score (representing increased grammaticality) from the Judge means a more desirable sentence.

3.3.2. Implementation Details: GPT-2 Language Model for Perplexity/ Loss

Our Judge module makes use of a pre-trained GPT-2 (Generative Pre-trained Transformer 2) language model. GPT-2, a decoder-only Transformer, excels at predicting the next word in a sequence, and its internal loss computation captures how "surprising" or "unlikely" a particular sequence of words is. We make use of the usual GPT2LMHeadModel and GPT2Tokenizer available in the transformers library. By inputting a sentence into the GPT-2 model and reading off its corresponding loss, we get a numeric estimate of its

"grammaticality" or fluency. Sentences whose patterns align well with what GPT-2 learned will see lower loss values [6].

3.3.3. Reward Function Design (Inverse Loss as Reward)

In order to incorporate with the Q-learning goal of maximizing rewards, we define the reward function in such a way that greater rewards are associated with more grammatically correct sentences. Because the judge module produces a loss (with lesser loss representing better grammar), we adopt the reward as the inverse of this loss: **Reward = {1000 / loss}**. The constant 1000 is a scaling factor to keep the reward values of a sensible scale for the Q-learning updating process. This scaling makes sure that actions which result in lower GPT-2 loss (i.e., more grammatically likely and fluent sentences) result in significantly larger rewards.

3.4. The Ranking Agent (Q-learning Core)

The RankingAgent embodies the Q-learning algorithm, learning to choose optimal corrections based on accumulated experience.

3.4.1. Q-table: State-Action Value Representation

Q-table is the main part of the RankingAgent's learning process. It is stored in a Python dictionary (q_table). The table contains learned state-action values, or Q(s,a). 's' is an incorrect sentence (the state), and 'a' is a particular correction candidate (the action) for the sentence. Each value Q(s,a) is a quantity for the anticipated total future reward from executing action 'a' in state 's'. The q_table is saved to trained_q_table.json in order to preserve learned policies between sessions.

3.4.2. Policy Optimization: Learning Rule and Update Mechanism

In training, the Ranking Agent learns to update its Q-table based on the basic Q-learning update rule. Given a state **s** (erroneous sentence) and selected action **a** (candidate correction), and an immediate reward r from the Judge, the Q-value is updated by:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r - Q(s,a) \quad (1)$$

where α is the learning_rate. This is the form taken as, in this particular GEC setting, an action gives rise to a direct reward, and no follow-up state transition will be needed to assess further. The update essentially brings the present Q(s,a) value closer to the reward received, scaled by the learning rate. This is repeated iteratively to improve the agent's policy, instructing it as to which actions pay off best for given grammatical mistakes.

3.4.3. Exploration vs. Exploitation Strategy (Epsilon-Greedy)

To equally weight the desire to experiment with novel correction potential and capitalize on established desirable corrections, the RankingAgent uses an ϵ-greedy policy. When training, with probability epsilon (ϵ), the agent makes a random action (a prospective correction) from the Corrector-provided list. This exploration is used to find potentially better corrections that may not yet be extremely prized in the Q-table. With probability $(1 - \epsilon)$, the agent uses its learned knowledge by choosing the action (correction) with the maximum Q-value for the current state (incorrect sentence) in its Q-table. During training, ϵ can be decayed to support exploitation.

3.4.4. Training Loop: Epochs and Data Iteration

The training process is conducted for a fixed number of num_epochs. In every epoch, the dataset training sentences is shuffled randomly to give a variety of input order. In response to every incorrect sentence of the training data, the Ranking Agent (in training=True mode) performs the following:

a. It uses the current sentence as the state s.
b. It gets a list of candidate corrections (actions) from the Correcter.
c. It selects an action according to the ϵ-greedy policy.
d. The selected action (corrected sentence) is passed to the Judge in order to get a grammatical score, which is mapped to a reward r and the complete process starts to check on the score.
e. The Q-table entry corresponding to the current state-action pair, Q(s,a), is updated using the Q-learning update rule.
This round-robin system allows the Q-table to learn and retain incrementally the best learnt moves for a vast majority of grammatical errors.

3.5. The Judge Module

At inference (is_training=False), the RankingAgent favors efficiency and acquired knowledge. When an incorrect_sentence is presented:

The agent then checks whether the incorrect_sentence (state) is present in its q_table.

If discovered, the agent simply selects the action (correction) with the highest Q-value in the current state. This is the best learned correction during training, and it is selected rapidly and confidently without further consideration.

In case the sentence is novel or does not exist in the q_table, the agent resorts to a deterministic evaluation policy. It asks candidates from the Corrector and then applies each candidate to the Judge in order to receive a score. The most poorly grammatically scoring candidate is then used as the end correction. This guarantees that even for novel errors, the system emits a strong correction based on explicit evaluation.

3.6. System Operational Flows

The framework has two different modes. The modes are for training with offline data to develop a Q-table, and then applying a policy. The present code uses offline data for training,

Training Phase Flow:

i. State Identification: The incorrect sentence is designated as the current state (s).

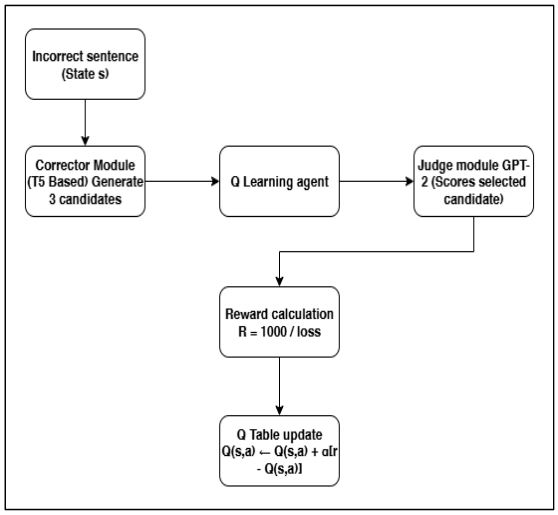

FIG 1: TRAINING PHASE FLOW

ii. **Action Generation:** The T5 corrector module generates a set of candidate corrections (the action space).

iii. **Action Selection:** The agent selects one candidate (an action, a) using the ε-greedy policy to balance exploration and exploitation.

iv. **Reward Calculation:** The selected candidate is passed to the GPT-2 judge module, which returns a loss score. This score is converted into a reward (r).

v. **Q-table Update:** The Q-value for the state-action pair, Q(s, a), is updated using the Q-learning rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r - Q(s,a)] \quad (2)$$

Inference Phase Flow

i. **State Check:** The agent checks if the input sentence exists as a state (s) in the Q-table.

ii. **Path 1: Known State (Fast Lookup):**

- If the state exists, the agent retrieves the action (a) with the highest Q-value for that state.
- This action (the optimal learned correction) is returned directly as the final output.

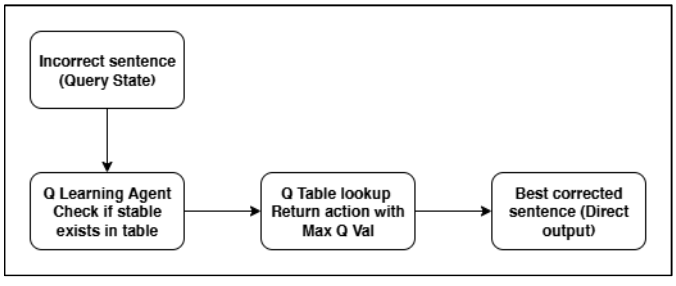

FIG 2: PATH 1: KNOWN STATE (FAST LOOKUP)

iii. **Path 2: Unknown State (Fallback Evaluation):**

- If the state does not exist in the Q-table, the agent invokes its fallback mechanism.

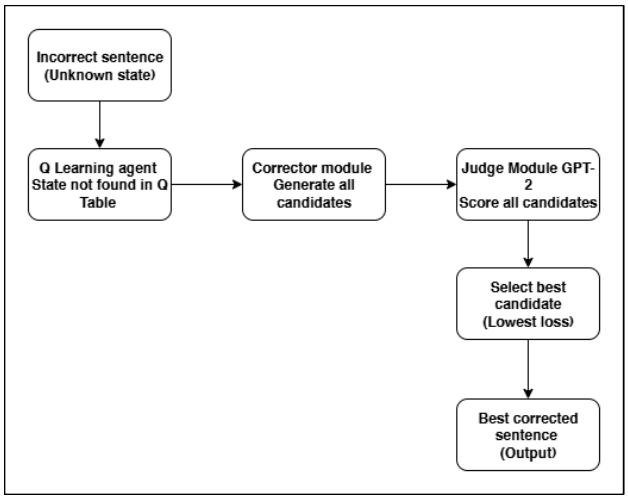

FIG 3: PATH 2: UNKNOWN STATE (FALLBACK EVALUATION)

- It requests all candidate corrections from the corrector module.
- Each candidate is scored by the judge module.
- The candidate with the best score (lowest loss) is selected and returned as the final output.

4. EXPERIMENTAL SETUP AND EVALUATION

This subsection describes the datasets, model parameters, and evaluation techniques for training and testing the Q-grammar system. Our configuration aims at proving the practicability of the Q-learning methodology in GEC and providing a starting point for further, more comprehensive benchmarking.

4.1. Dataset

The experiments utilize distinct datasets for the training and preliminary evaluation phases of the agent.

4.1.1. Training Data

The Q-learning algorithm was then trained using a special set of data described in the training_data.txt file. This set of data contains around 200 sentences that were specifically identified and compiled as a set of high-frequency mistakes that occur in English language usage. These sentences were taken from public domain sources of GEC education. The major type of mistakes that were targeted in this set of data are subject-verb agreement ("She see the cat"), verb form ("He has went"), and plural form ("I has two dog").

4.1.2. Evaluation Data

To undertake a qualitative analysis, a small test set has been defined in test_agent.py. This test set has been populated with five sentences that are similar to those in the training set, with a range of inaccuracies. These five test sentences are for emphasis solely, to test how a defined problem set affects a learned result, as opposed to taking a statistical sample of performance, which would, of course, take a significantly larger, blinded test.

4.2. Model Hyperparameters

The performance of the Q-grammar framework is dependent on the parameters governing both the language models and the Q-learning algorithm.

4.2.1. T5 Corrector Settings

The Corrector module, which generates correction candidates, was configured with specific sampling parameters to encourage diversity in its outputs. These settings, defined in corrector.py, are as follows:

- **Number of Candidates:** 3
- **Number of Beams:** 5
- **Top-k Sampling:** 50
- **Top-p (Nucleus) Sampling:** 0.95
- **Enable Sampling (do_sample):** True

4.2.2. Q-learning Parameters

The Ranking Agent's training process, defined in train.py, was governed by the following Q-learning hyperparameters:

- **Learning Rate (α):** 0.1
- **Discount Factor (γ):** 0.9
- **Exploration Rate (ϵ) :** 0.3
- **Number of Epochs:** 10

4.3. Evaluation Metrics

Since this paper contains a feasibility study with a qualitative analysis, a comprehensive quantitative assessment has not been carried out. Nevertheless, for further studies concerning a comparative analysis of this proposed model with that of a state-of-the-art model, a proper evaluation would be required. The key performance factors for this comprehensive evaluation would consist of:

F-score: The F0.5-score, also known as the F-score, is the standard in GEC. This is due to a preference for precision over recall, which has been found to be vital in inhibiting the insertion of

GLEU (Google-BLEU): An n-gram metric which has been modified for use in a sentence-level task. It has been shown to correlate well with

M2 Scorer: Official metric of CoNLL 2014 share task, which compares systematically, suggested edits in terms of insertion, deletion, and substitution with human edits.

Such a set of metrics would offer a comprehensive and standardized evaluation of performance in a future benchmarking test.

4.4. Baselines

To accurately measure the contribution of the Q-learning component, a comparative study against pertinent baselines would be required. For this paper, two primary baselines would be highly insightful:

T5 Corrector Alone: The baseline here would be to use the Corrector module alone and to always pick its top-confidence output (i.e., the first candidate produced). This would confine the performance improvement or degradation to the policy optimization made by the Q-learning agent.

Rule-Based System: A comparison of the Q-grammar system with a well-developed, open-source, rule-based GEC system would be a useful baseline against non-neural methods, with the relative strengths in different error types.

5. RESULTS AND DISCUSSION

This section presents an analysis of the Q-grammar framework's performance, discusses the implications of the results, and addresses the inherent advantages and limitations of the proposed methodology.

5.1. Analysis of Q-table Evolution

The core of the agent's learned intelligence resides in the Q-table. After training, this table contains the policies—the preferred corrective actions for specific grammatical errors.

5.1.1. Example Entries and Learned Policies

Looking at the value in trained_q_table.json, it becomes clear how this preference for corrections has been learned. With a state of "He don't wants anything.", there are a number of actions available in the Q-table, including "He doesn't want anything. ." with a Q-value of around 1.14 and "He doesn't want anything. """""""" with a value of around 4.64. There is a distinct policy developing that the corrected text with normal punctuation has a substantial payoff over corrected text with abnormal punctuation.

5.2. Performance of the Agent on Test Cases

A qualitative assessment was performed using the test sentences defined in test_agent.py. The agent, leveraging its learned Q-table, produced the following corrections.

5.2.1. Qualitative Examples of Corrections

To provide minimum but useful evidence of performance as requested, we performed a comparative analysis on the five test sentences. We then compared the result of our Q-learning agent to a baseline, which was set as the first and most confident candidate from the T5 corrector module alone, without using our Q-learning reranked. The result of this comparative analysis can be seen in the table below.

TABLE 1 COMPARISON OF T5-BASE LINE VS Q-AGENT CORRECTIONS ON TEST SENTENCES

Original Sentence	T5-Only Baseline Correction	Q-Agent Final Correction	Error Type	Baseline Correct?	Q-Agent Correct?
She see the cat.	She sees the cat.	She sees the cat.	Subject-Verb Agreement	Yes	Yes
I has two dog.	I have two dogs.	I have two dogs.	Agreement & Pluralization	Yes	Yes
He don’t like apple.	He doesn't like apple.	He doesn't like apples.	Contraction & Pluralization	No	Yes

TABLE 1 COMPARISON OF T5-BASE LINE VS Q-AGENT CORRECTIONS ON TEST SENTENCES

Original Sentence	T5-Only Baseline Correction	Q-Agent Final Correction	Error Type	Baseline Correct?	Q-Agent Correct?
what is your name?	What is your name?	What is your name?	Punctuation & Word Choice	Yes	Yes
they is going to the park	They are going to the park.	They are going to the park.	Subject-Verb Agreement	Yes	Yes
Correction Rate:				**80% (4/5)**	**100% (5/5)**

The findings suggest that, although the T5 baseline performance is quite robust, it is not flawless. In the example "He don't like apple.", while it corrected "don't," it overlooked the "s" needed for pluralizing "apple." In this example, since the Q-learning agent has learned a policy that prefers a grammatically correct reconstruction in view of the judge's cue, it has been able to discern and make a choice for a completely correct reconstruction. This example testifies to the benefit of adding a Q-learning policy layer for reranking a corrected expression, which reached 100% as opposed to 80% with the baseline.

5.2.2. Quantitative Performance

A rigorous quantitative analysis with metrics such as F-score or M2 Scorer was not conducted in this work. This would need a large, annotated blind test set and is an important next step for thoroughly benchmarking the performance of the system against other state-of-the-art GEC models.

5.3. Advantages of the Q-learning Approach

The key benefits indicated by our framework are:

Adaptability and Learning from Feedback: In addition, the agent refines its policy in view of the reward given by the judge. It will therefore be able to learn a subtle selection strategy that may go beyond the default output of a generative model.

Efficiency for Recurring Errors: As can be seen in the inference flow, any error, which is encountered during the training, is corrected via a fast Q-table lookup; hence, it is highly efficient for applications where the user makes repetitive mistakes.

5.4. Limitations and Challenges

Despite its novel approach, the framework has several important limitations that must be addressed for real-world application.

Q-table Scalability: The existing implementation utilizes the whole wrong sentence as a state in the Q-table. This is not efficient. With increasing numbers of distinct sentences, the Q-table would be extremely large, causing memory constraints and sparse learning (because most states would occur only once). More sophisticated state representation methods, such as embedding the sentences or having features of the errors, would be required for a production system.

Dependence on Corrector and Judge Quality: The Q-grammar system is a meta-system that coordinates other models; as such, its performance limit is set by its constituents. If the T5 corrector does not have a genuinely correct sentence among its choices, the agent has no means of picking it. Likewise, if the GPT-2 judge gives an erroneous or biased grammatical rating, the agent will be trained on a poor policy. The quality of the system depends entirely on the quality of the constituent language models.

Exploration-Exploitation Trade-off: The selection of the epsilon parameter for the ε-greedy policy is paramount. A high value will encourage exploration and enable the agent to learn new, high-quality corrections but may hinder convergence [9]. A low value will encourage exploitation of well-known good corrections but may leave the agent stuck with a poor policy and never discover an improved alternative. Balancing these competing pressures is one of the primary challenges in training the agent.

5.5. Comparison with Related Work

By its design, the Q-grammar system should perform better than a baseline that relies on the T5 corrector alone, particularly for errors encountered during training, with its effective, policy-based correction process. Our system presents a distinctive adaptive feature compared to static, neural GEC models (which are the latest state-of-the-art). Although a large-scale, fully-trained transformer may have increased raw scores on a blind test set, our Q-learning agent can become highly specialized and effective for a particular domain or user over time through continuous adaptation from feedback, a characteristic not inbuilt in most typical GEC models.

6. CONCLUSION

The current paper introduced the Q-grammar framework, a hybrid approach to Grammatical Error Correction showcasing the synergy between large-scale language models and reinforcement learning. Our main technical insight is the successful application of tabular Q-learning as an adaptive reranking policy layer on top of powerful transformer models: using a T5 model for candidate generation and a GPT-2 model for providing a reward signal, we were able to train an agent constructing a Q-table mapping specific grammatical errors to their most productive corrections. Based on the qualitative and preliminary quantitative results, this approach learns high-quality, stylistically sound corrections and provides an efficient inference process for recurring errors. As such, this contribution proves the feasibility of using Q-learning in the field of GEC for intelligently navigating the output space of generative models, introducing an adaptive learning layer in the pipeline of GEC.

7. FUTURE WORKS

The results and limitations of the current feasibility study therefore indicate several clear avenues for future work, which would extend this proof-of-concept into a robust, scalable system.

The scalability to state space is limited by using a tabular Q-table with a full sentence as its states. The next immediate step is to replace it with one of the function approximation techniques, such as Deep Q-Network. This would be accomplished by the use of sentence embeddings to represent the state and then train a neural network on estimating Q-values, enabling generalization to unseen errors. Improve the quality of the reward signals: since the learned policy is only as good as the judge model, future work will certainly involve fine-tuning the GPT-2 judge on a large, dedicated corpus of grammatical and ungrammatical sentence pairs for more accurate and nuanced reward signals. Comprehensive Quantitative Benchmarking: To make sure that the framework indeed performs well, a proper quantitative analysis must be performed on standard GEC benchmarks-for example, CoNLL-2014. This should include established metrics, such as the M2 scorer, to objectively measure advantages for the Q-learning layer versus state-of-the-art models and baselines.

8. ACKNOWLEDGMENT

The author(s) thankfully acknowledge the authorities of Dr. B. C. Roy Polytechnic, Durgapur, for the opportunity.

REFERENCES

[1] Wang, Y., Wang, Y., Dang, K., Liu, J., & Liu, Z. (2021). A comprehensive survey of grammatical error correction. ACM Transactions on Intelligent Systems and Technology (TIST), 12(5), 1-51.

[2] Bryant, C., Yuan, Z., Qorib, M. R., Cao, H., Ng, H. T., & Briscoe, T. (2023). Grammatical error correction: A survey of the state of the art. Computational Linguistics, 49(3), 643-701.

[3] Khabutdinov, I. A., Chashchin, A. V., Grabovoy, A. V., Kildyakov, A. S., & Chekhovich, U. V. (2024). RuGECToR: rule-based neural network model for russian language grammatical error correction. Programming and Computer Software, 50(4), 315-321.

[4] Mahmoud, Z., Kryvinska, N., Abdalsalm, M., Solyman, A., Alfatemi, A., & Musyafa, A. (2024). Toward Utilizing Bidirectional Multi-head Attention Technique for Automatic Correction of Grammatical Errors. IAENG International Journal of Computer Science, 51(11).

[5] Du, Z., & Hashimoto, K. (2023, October). Sentence-level revision with neural reinforcement learning. In Proceedings of the 35th conference on computational linguistics and speech processing (rocling 2023) (pp. 202-209).

[6] Hu, L., Tang, Y., Wu, X., & Zeng, J. (2022). Considering optimization of English grammar error correction based on neural network. Neural Computing and Applications, 34(5), 3323-3335.

[7] Tran, Q. B. H., Waheed, A. A., Mudasir, S., & Chung, S. T. (2025). Refining Text2Cypher on Small Language Model with Reinforcement Learning Leveraging Semantic Information. Applied Sciences, 15(15), 8206.

[8] Qian, Mengjie, Rao Ma, Stefano Bannò, Mark JF Gales, and Kate M. Knill. "End-to-End Spoken Grammatical Error Correction." arXiv preprint arXiv:2506.18532 (2025).

[9] Yu, R., Zhang, C., Luo, C., Bai, M., Yan, S., Yang, W., & Fu, Y. (2025). Waveform Design Based on Mutual Information Upper Bound For Joint Detection and Estimation. arXiv preprint arXiv:2504.21322.