Smruti: Grammatical Error Correction for Gujarati using LLMs with Non-Parametric Memory

Vrund Dobariya, Jatayu Baxi, Bhavika Gambhava, Brijesh Bhatt
Dharmsinh Desai University
vrund3626@gmail.com, {jatayubaxi.ce,bhavika.ce,brij.ce}@ddu.ac.in

Abstract

Grammatical Error Correction (GEC) is a fundamental task in Natural Language Processing that focuses on automatically detecting and correcting grammatical errors in text. In this paper, we present a novel approach for Gujarati GEC. Gujarati is an Indian language spoken by over 55 million people worldwide. Our approach combines a large language model with non-parametric memory modules to address the low-resource challenge. We have evaluated our system on human-annotated and synthetic datasets. The overall result indicates promising results for Gujarati. The proposed approach is generic enough to be adopted by other languages. Furthermore, we release a publicly available evaluation dataset for Gujarati GEC along with an adapted version of the ERRANT framework to enable error-type-wise evaluation in Gujarati.

1 Introduction

Grammatical Error Correction (GEC) is necessary not only for enhancing the quality of text but also for applications such as language learning and automated writing evaluation. For instance, Gujarati typing systems, such as Google's GBoard, take Roman transliterations from the user as input and convert them into Gujarati script based on dictionary lookup and word frequency. This approach is prone to grammatical errors, and an accurate GEC system can be beneficial in this context. GEC can be viewed as a form of machine translation that transforms the input text by correcting the errors and producing a corrected output (Yuan and Briscoe, 2016). Over the years, the field has advanced considerably, evolving from rule-based approaches and statistical classifiers to statistical machine translation (SMT), neural machine translation (NMT) systems, and most recently, the transformer-based models (Wang et al., 2020). These approaches typically require a large amount of labeled data, which is not feasible for low-resource languages such as Gujarati. The rich morphology and complex lexicography of Gujarati (Patel and Patel, 2015) further complicate the task.

We classified grammatical errors into the categories as described in Figure 1.

1. Disagreement Error
✗ તેઓ રમતો હતો. (teo ramato hato.)
"They was playing."
✓ તેઓ રમતા હતા. (teo ramatā hata.)
"They were playing."

2. Word Order Error
✗ તમે મારી ચાલો સાથે. (tame māri cālo sāthe.)
"You come me with."
✓ તમે મારી સાથે ચાલો. (tame māri sāthe cālo.)
"You come with me."

3. Morphological Errors
- Inflectional:
  ✗ ગઈકાલે વરસાદ પડે. (gaīkāle varasād pade.)
  "It rains yesterday."
  ✓ ગઈકાલે વરસાદ પડ્યો. (gaīkāle varasād padyo.)
  "It rained yesterday."
- Derivational:
  ✗ સિંહણી (sinhaṇī) → "lioner"
  ✓ સિંહણ (sinhaṇ) → "lioness"

4. Orthographic Errors
- Spelling:
  ✗ સ્મૃતી (smṛtī) → "remembrence"
  ✓ સ્મૃતિ (smṛti) → "remembrance"
- Punctuation:
  ✗ તમે શું કરો છો (tame śu karo cho)
  "What are you doing"
  ✓ તમે શું કરો છો? (tame śu karo cho?)
  "What are you doing?"

Figure 1: Examples of grammatical errors with transliterations and analogous English examples

The emergence of large language models (LLMs) opened up new avenues for low-resource languages. Approaches such as advanced prompting techniques, synthetic data generation using

LLMs, and information retrieval reduce the need for labeled data.

The key contribution of this work is following:

1. We present a system for Gujarati grammatical error correction that incorporates non-parametric memory with LLM. We compare its performance against a fine-tuned transformer-based model trained on synthetic data.[1]

2. We release a resource suite for Gujarati GEC, including an adaptation of the ERRANT toolkit for detailed error-typewise evaluation[2], a synthetic dataset consisting of 5,000 samples, and a high-quality human-annotated dataset consisting of 300 samples for evaluation purpose. Moreover, we release 10,000 gold-standard Gujarati sentences.

2 Related Work

GEC research has evolved through multiple stages. Early work in GEC focused on classifier-based methods using manually designed features for specific error types (Lee, 2004). With the availability of annotated corpora like NUCLE (Dahlmeier et al., 2013), monolingual translation and Statistical Machine Translation (SMT) were applied to solve the problem. (Junczys-Dowmunt and Grundkiewicz, 2014). Later, Neural Machine Translation (NMT) approaches using RNNs (Yuan and Briscoe, 2016), CNNs (Kalchbrenner and Blunsom, 2013), and Transformers (Grundkiewicz and Junczys-Dowmunt, 2018) started dominating. Recently, edit-based methods have also gained popularity. GECToR (Omelianchuk et al., 2020) frames GEC as a sequence tagging problem. Seq2Edits (Stahlberg and Kumar, 2020), on the other hand, frames it as a sequence-to-sequence edit generation task (Bryant et al., 2023).

Large language models (LLMs) have achieved strong results with zero-shot and few-shot Chain-of-Thought settings (Fang et al., 2023). Long-term memory, inspired by humans (Wu et al., 2025; Zhang et al., 2024), allows the LLM to store, update, and reuse useful knowledge over time. Advancements in non-parametric memory mechanisms for LLMs, such as RAG for long-term memory (Lewis et al., 2020) and ReAct for short-term memory (Yao et al., 2023), have gained popularity across a wide range of domains (Wu et al., 2025). The design of our system is inspired by the long-term memory frameworks proposed in Wang et al., 2025 and Gutiérrez et al., 2025.

A widely adopted strategy to mitigate data scarcity is to train or fine-tune an LLM or transformer-based model on a synthetic dataset. This approach has been explored for several languages such as Hindi (shares linguistic features with Gujarati) (Sharma and Bhattacharyya, 2025), Czech (Náplava and Straka, 2019), Indonesian (Musyafa et al., 2022), Spanish (Kubal and Nagvenkar, 2025), and Chinese (Fan et al., 2023). Apart from this, Li et al., 2025 uses separately retrieved grammatically correct and erroneous sentences to guide the LLM in generating responses.

Gujarati GEC remains an underexplored area; existing research has primarily focused on spelling correction. Patel et al., 2021 employed string matching techniques for spell correction, followed by a rule-based spell checker proposed in Gondaliya et al., 2022. Additionally, Panchal and Shah, 2024 applied Norvig's algorithm for Gujarati spelling correction.

3 Dataset

To the best of our knowledge, there is no standardized labeled dataset available for Gujarati to solve the problem of GEC. We have created a dataset for GEC; we find collecting correct sentences and introducing errors in them relatively easier. Publicly available unlabeled datasets such as IndicCorp (Doddapaneni et al., 2023) and CC-100 (Conneau et al., 2020; Wenzek et al., 2020) contain data from news articles and web crawling. Hence, we created a pool of 2,04,169 unique sentences from the books of Gujarati literature to ensure higher linguistic quality and diversity (Gujarati Wikisource contributors). The text taken from the book was divided into sentences by using the `GPT-4o-mini` LLM. Then, a Python script was used to create the dataset splits described in Table 1. We also ensure that there is no overlap of data among the splits.

Gold Sentences are verified, grammatically correct Gujarati sentences out of the pool. These sentences are used for populating memory module M_1 as shown in Section 4.2.

Erroneous Sentences are used to evaluate the impact of M_2 memory module on the overall performance of the system. This split is created by introducing errors in correct sentences using a rule-based method as described below:

[1]https://github.com/VGD3626/Smruti-GEC-for-Gujarati

[2]https://github.com/VGD3626/ERRANT-for-Gujarati

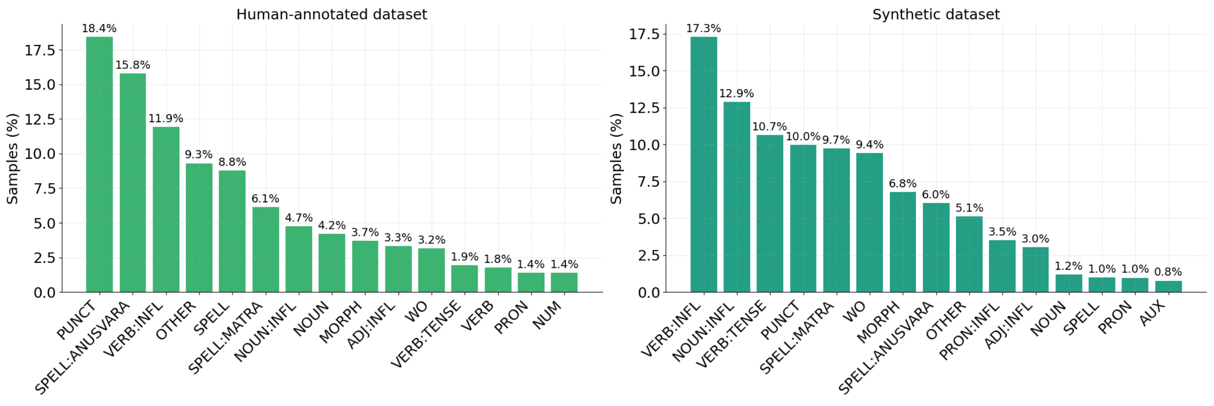


Figure 2: Distribution of various error types in evaluation set

Dataset	**Format**	**#Samples**	**#Errors**	**#Tokens**	**Purpose**
Gold Sentences	Correct Sentences	10,000	–	150,898	Populating M_1
Erroneous Sentences	Incorrect Sentences	10,000	8,216	165,463	Populating M_2
Human Evaluation Set	Correct–Incorrect Tuples	300	570	7,428	Evaluation
Synthetic Evaluation Set	Correct–Incorrect Pairs	5,000	6,092	164,666	Evaluation

Table 1: Summary of the datasets used in our experiments. The symbol '#' indicates the count. The table provides the format, total number of samples, total errors, and total tokens for each dataset.

Let S be the randomly selected sentence from a set of grammatically correct Gujarati sentences. Then, one of the following four operations is applied to S, with respective probabilities P_1, P_2, P_3, and P_4:

- *Punctuation Error*: Select a punctuation mark $p \in S$ uniformly at random and either delete it or replace it with an alternative punctuation mark p', where $p' \neq p$.
- *Morphological Error*: Select a word $w \in S$, apply a rule-based Gujarati stemmer to obtain its stem w_s, and attach a randomly chosen suffix σ from the set of Gujarati suffixes Σ. If the resulting word $w' = w_s + \sigma$ is not found in the lexicon $\mathcal{L}$, discard the current sentence and repeat the process on another sentence until a word $w' \in \mathcal{L}$ is generated, to introduce an inflectional or derivational error.
- *Word Order Error*: Randomly choose two adjacent non-punctuation words $w_i, w_{i+1} \in S$ and swap their positions to generate a syntactic error.
- *Orthographic Error*: Select a word $w \in S$, and either insert/delete an *anusvāra* or modify a *mātrā* within w to generate an orthographic error.

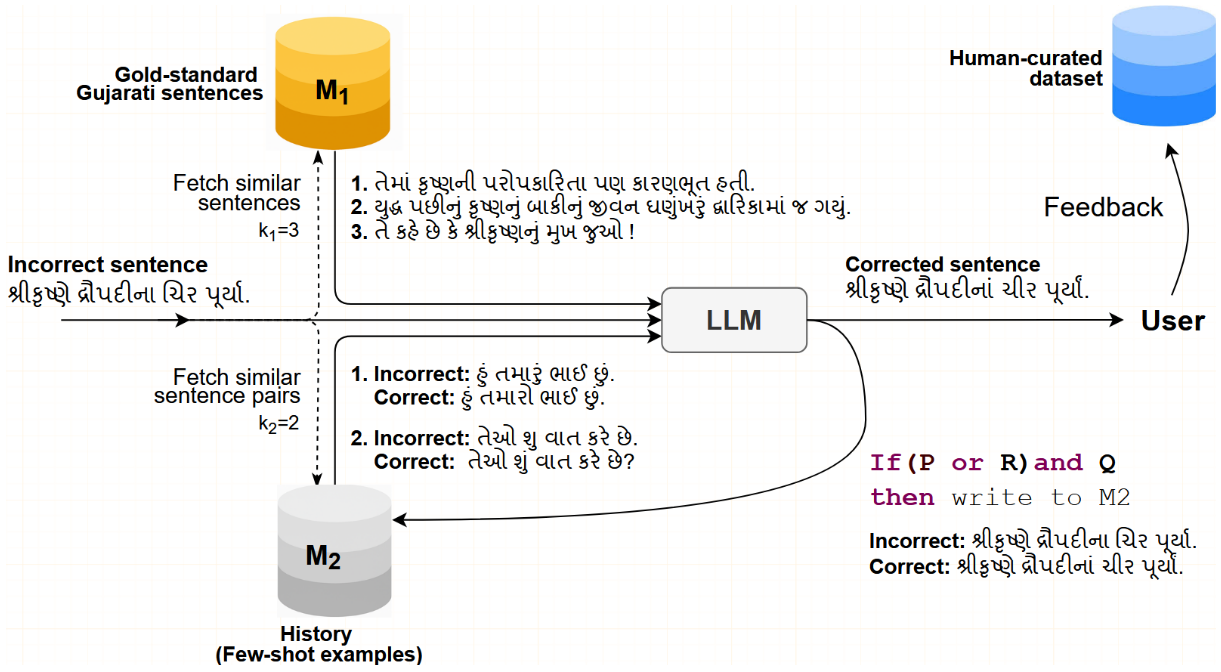
We limit the maximum number of errors per sentence to 3 to maintain sentence interpretability. We obtained a less-skewed error-type distribution by setting the probabilities as $P_1 = 0.10$, $P_2 = 0.65$, $P_3 = 0.10$, and $P_4 = 0.15$.

3.1 Evaluation Set

We evaluated the system on human-annotated and synthetic datasets. The synthetic dataset, comprising 5,000 samples, was generated using the error generation method described above under the same settings.

The human-annotated dataset, on the other hand, was generated with the process of dictation. Hence, this dataset contains transcription errors. Two native Gujarati individuals volunteered to write 1,000 sentences were taken from the pool, and these sentences were given to a linguist, a different individual, to identify the errors. As identified by the linguist, there were 366 sentences with errors. From these 366 sentences, 300 sentences were selected randomly with the help of a Python code in order to achieve a perfect figure of 300. Then, the linguist corrected these 300 sentences. The linguist was allowed to annotate an incorrect sentence with more than one correct sentence.

Figure 2 presents the error-type distribution in the evaluation set based on our Gujarati ERRANT


Gold-standard Gujarati sentences
M1
Human-curated dataset
Fetch similar sentences k1=3
Feedback
Incorrect sentence
LLM
Corrected sentence
User
Fetch similar sentence pairs k2=2
M2
History (Few-shot examples)

Incorrect sentence
શ્રીકૃષ્ણે દ્રૌપદીના ચિર પૂર્યા.

1. તેમાં કૃષ્ણની પરોપકારિતા પણ કારણભૂત હતી.
2. યુદ્ધ પછીનું કૃષ્ણનું બાકીનું જીવન ઘણુંખરું દ્વારિકામાં જ ગયું.
3. તે કહે છે કે શ્રીકૃષ્ણનું મુખ જુઓ !

Corrected sentence
શ્રીકૃષ્ણે દ્રૌપદીનાં ચીર પૂર્યાં.

1. **Incorrect:** હું તમારું ભાઈ છું.
   Correct: હું તમારો ભાઈ છું.
2. **Incorrect:** તેઓ શુ વાત કરે છે.
   Correct: તેઓ શું વાત કરે છે?

```
If(P or R)and Q
then write to M2
```

Incorrect: શ્રીકૃષ્ણે દ્રૌપદીના ચિર પૂર્યા.
Correct: શ્રીકૃષ્ણે દ્રૌપદીનાં ચીર પૂર્યાં.

Figure 3: System Architecture Diagram

adaptation, as described in Section 5. Since the ERRANT adaptation relies on limited linguistic resources, the distribution may include some noise and should be interpreted as an approximate estimate.

4 Proposed System

As shown in Figure 3, our system consists of three major components:

1. A large language model (LLM) for correcting the sentence.
2. An embedding model for generating sentence-level embeddings.
3. Memory modules M_1 and M_2 for storing necessary information.

Initially, M_1 is populated with gold-standard sentences. M_2 is empty initially and is updated as the system performs corrections. The operational aspects of our system include sentence correction, memory management, and hyperparameter tuning.

4.1 Sentence Correction

The error correction is performed using the following steps:

1. The user provides an incorrect sentence as input. Embedding model generates an n-dimensional sentence-level embedding for the input.
2. k_1 and k_2 records are retrieved from memory modules M_1 and M_2 respectively. These retrieved records are included in the prompt along with task-specific instructions.
3. The LLM generates a corrected version of the sentence based on the constructed prompt. The system then waits for human feedback on the generated correction.
4. The human feedback is considered positive, either because the user confirms that the LLM's correction is accurate or the user manually provides the correct version, the system adds the *(incorrect, corrected)* sentence pair to the human-curated dataset.
5. Additionally, the sentence pair *(incorrect, corrected)*, along with an embedding for the incorrect sentence, is stored in memory module M_2 if the condition for writing into M_2 is satisfied.

4.2 Memory Management

Memory plays an important role in the overall performance of the system. The memory modules M_1 and M_2 function as long-term memory, while the prompt serves as a form of short-term memory (Wu et al., 2025).

The purpose of Memory module M_1 is to provide context for generating the response. To preserve M_1's integrity and avoid introducing errors, it is kept as a read-only memory. M$_2$ gives the system the ability of non-parametric continual learning (Gutiérrez et al., 2025). Corrections stored by M$_2$ act as few-shot examples for future corrections.

4.2.1 Memory Read

Let $q \in \mathbb{R}^n$ be the sentence embedding corresponding to the incorrect sentence (user input), and let $m_i \in \mathbb{R}^n$ denote the embedding stored for the i-th record in memory.

A semantic search is performed by computing the distance between q and each m_i. The top-k most similar records are retrieved based on this distance.

We use cosine-distance as a distance matrix in our experiments. The distance between the incorrect sentence and a memory record is defined as:

$$d(q, m_i) = 1 - \frac{q \cdot m_i}{\|q\| \, \|m_i\|} \tag{1}$$

4.2.2 Memory Write

Among the two memory modules, only M$_2$ gets updated over time. The following considerations are taken into account while writing into M$_2$:

1. If the correction generated by the LLM is incorrect, storing it provides no value and may degrade the quality of memory.
2. If a similar correction already exists in memory, adding another similar record can lead to redundancy.
3. If the correction receives positive feedback from the user, it is considered valuable and is stored in memory.

Let q denote the incorrect sentence (user input) and r be the response generated by the LLM. Let $m_1^{(1)}, m_2^{(1)}, \ldots, m_{k_1}^{(1)}$ represent the embeddings of the top-k_1 records retrieved from memory $\mathtt{M}_1$, and $m_1^{(2)}, \ldots, m_{k_2}^{(2)}$ represent the embeddings of the top-k_2 records retrieved from memory $\mathtt{M}_2$. Consider the logical statements P, Q, and R :

$$P : \quad \frac{1}{k_1} \sum_{i=1}^{k_1} d(q, m_i^{(1)}) \leq \delta_1, \quad \text{where } m_i^{(1)} \in \mathbb{R}^n$$

The statement P follows a hypothesis that if the average distance between embeddings of the incorrect sentence and the top-k_1 retrieved records from M$_1$ is below a threshold δ_1, then the likelihood of generating an accurate correction is higher.

$$Q : \quad \min_{1 \leq i \leq k_2} d(r, m_i^{(2)}) \geq \delta_2, \quad \text{where } m_i^{(2)} \in \mathbb{R}^n$$

The statement Q ensures that even the most similar record in M$_2$ is at least δ_2 away from the LLM's response to avoid redundancy in memory.

$$R = \begin{cases} \text{TRUE} & \text{if the user gives positive feedback} \\ \text{FALSE} & \text{otherwise} \end{cases}$$

Based on these statements, the final condition for writing into memory module M$_2$ is given by,

$$\left(P \vee R\right) \wedge Q$$

If this condition is satisfied, a record consisting of the incorrect sentence, the corrected sentence, and the embedding for the incorrect sentence is stored in M$_2$.

4.3 Hyperparameter Tuning

The system requires the following hyperparameters to be tuned for achieving a good performance with time:

k_1, k_2: Number of records (gold-standard sentences) retrieved from M$_1$ and number of records (few-shot examples) retrieved from M$_2$, respectively.

δ_1: The value of δ_1 was initially set to a low threshold and gradually increased as the number of examples stored in M$_2$ grew. As the number of records in the memory module M$_2$ increases, the variety of records within M$_2$ should also increase, since the parameter δ_2 prevents redundancy in this module. We hypothesize that this increased variety improves the relevance of the records retrieved from M$_2$. As more relevant few-shot examples are included in the prompt, the system should be able to handle increasingly difficult examples.

Furthermore, as the quality and diversity of the records stored in M$_2$ improve, the LLM should be able to correct difficult examples even when the records retrieved from memory module M$_1$ are semantically distant from the user input. Consequently, δ_1, which ensures the relevance of the records retrieved from M$_1$, can be made less stringent—that is, it can be increased.

This can be illustrated with the following example.

Suppose that the few-shot examples retrieved from M_2 are of very high quality, but the input sentence is quite semantically distant from those retrieved from M_1, and the logical statement P (Section 4.2) becomes false. This results in the memory write condition being evaluated as false. However, since the few-shot examples retrieved from M_2 are of high quality, the likelihood of producing an accurate correction remains high and, therefore, these examples should be stored in M_2.

We employed three heuristics to update δ_1. The first heuristic kept δ_1 constant, which contradicted the hypothesis. The second heuristic increased δ_1 linearly with the number of stored examples. The third heuristic increased δ_1 exponentially. Let t be the total number of records stored in M_2, and let $\delta_1^{(0)}$ be the initial value of δ_1. The value of $\delta_1^{(t)}$ in each case is defined as:

$$\delta_1^{(t)} = \delta_1^{(0)} \qquad \text{(Constant)}$$

$$\delta_1^{(t)} = \delta_1^{(0)} + \alpha t \qquad \text{(Linear)}$$

$$\delta_1^{(t)} = \delta_1^{(0)} + \alpha \times \left(e^{\alpha t} - 1\right) \qquad \text{(Exponential)}$$

The value of α, which controls the scaling of the update function, was set based on the total number of sentences the system was expected to correct during its operation.

δ_2: This threshold was initialized with a suitable value and kept constant throughout the operation of the system.

5 Experiments and Results

5.1 Evaluation Metrics

We use M^2 (max-match) score (Dahlmeier and Ng, 2012) for evaluating our system[3]. Since no public error annotation tools exist for Gujarati, we adapted the Error Annotation Toolkit (ERRANT) (Bryant et al., 2017). We used a rule-based stemmer, a rule-based lemmatizer backed by the Unimorph dataset (Batsuren et al., 2022) as a dictionary, and a transformer-based POS and morph model (Baxi et al., 2024) for implementation. We follow the error classes mentioned in Bryant et al., 2017. While the output can contain misclassifications, it enables coarse-grained error-type analysis for Gujarati. Our implementation incorporates two Gujarati-specific error types to further classify spelling errors: `SPELL:MATRA` for *mātrā* related errors (e.g., દીવસ- *dīvasa* $\rightarrow$ દિવસ- *divasa*) and `SPELL:ANUSVARA` for *anusvāra* related errors (e.g., અબર- *abara* $\rightarrow$ અંબર- *aṁbara*).

[3]https://github.com/nusnlp/m2scorer

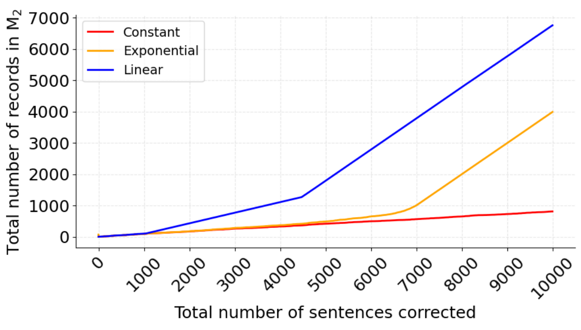


Figure 4: Size of M_2 with number of sentences corrected for three heuristics

5.2 Setup and Results

We conducted experiments using the `GPT-4o-mini` model. Sentence-level embeddings were generated using the multilingual `jina-embeddings-v3` model (Sturua et al., 2024). Zero-shot performance of `GPT-4o-mini` model on the evaluation dataset was taken as a baseline. As shown in Table 2, Chain-of-Thought prompting gave a small improvement.

We continued experiments with Chain-of-Thought prompting by first incorporating the memory module M_1, which contained 10,000 gold-standard Gujarati sentences. We initialized $\delta_1^{(0)}$ to 0.3 and δ_2 to 0.1, and we arbitrarily set $k_1 = 5$ and $k_2 = 2$ to explore the best update strategy for δ_1. The system was then provided with 10,000 erroneous sentences for correction to populate M_2. As described in Section 4.3, δ_1 is updated using three different heuristics. To observe the system's behavior accurately, we relied primarily on the human-annotated evaluation set, as small changes might not be reliably captured on the synthetic dataset due to inherent noise. The observations are shown in Table 3. Observations indicate that an exponential increase in δ_1 gives the best results.

Figure 4 illustrates the size of M_2 with the number of sentences corrected. We observe that the system started storing most of the sentences into the memory in case of linear and exponential increase after hitting a certain value of δ_1; it is likely due to the behaviour of the embedding model.

After finalizing the update strategy for δ_1, we tuned the hyperparameters k_1 and k_2. We perform a grid search over the set $\{0, 1, 3, 5, 7\} \times \{0, 1, 3, 5, 7\}$ to observe the system's behavior

Experiment name	Human annotated			Synthetic		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$
`GPT-4o-mini` zeroshot (baseline)	46.67	31.41	42.53	29.64	28.05	29.30
`GPT-4o-mini` Chain-of-Thought	48.83	30.26	43.49	30.03	27.25	29.43
`GPT-4o-mini` with M_1 and M_2	**58.68**	**41.61**	**54.43**	32.46	31.22	32.20
Finetuned mT5-base	38.96	12.99	27.83	**74.62**	**50.03**	**67.94**

Table 2: Comparison of the system's performance with the baseline, Chain-of-Thought prompting and fine-tuned mT5 model performance on human-annotated and synthetic evaluation sets.

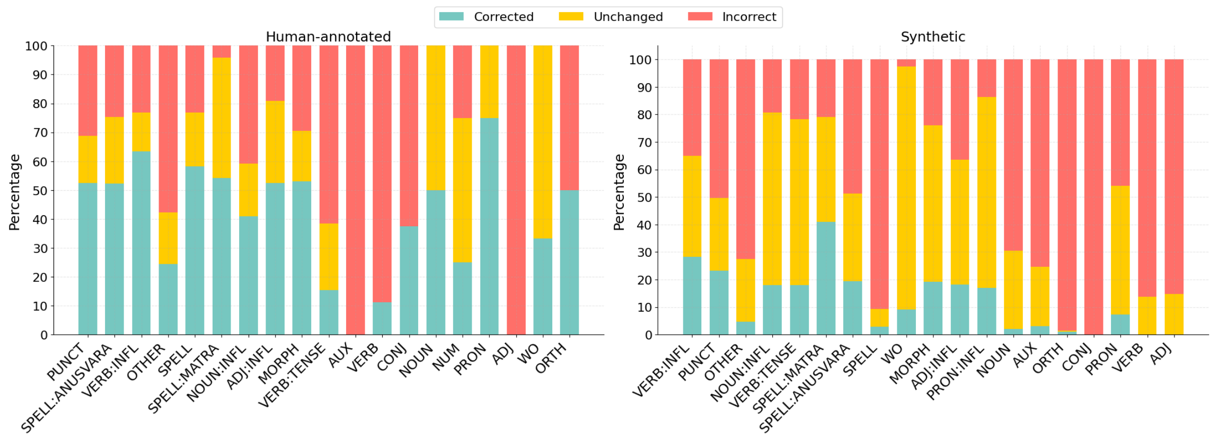


Figure 5: Performance of the system on various error types

Heuristic	$\delta_1^{(0)}$	α	P	R	$F_{0.5}$
constant	0.3	–	54.82	38.64	50.58
exponential	0.3	0.0035	**57.01**	**40.72**	**52.79**
linear	0.3	0.0001	55.36	40.72	51.65

Table 3: Comparison of heuristic strategies for updating δ_1, evaluated using M^2 score: Precision, Recall, and $F_{0.5}$.

across different combinations of k_1 and k_2 on the human-annotated evaluation set. We observe the maximum $F_{0.5}$ score of 54.43 for $k_1 = 7$ and $k_2 = 1$. Additionally, we found that setting k_1 to zero while varying k_2 improves the $F_{0.5}$ score by 3.52 points for $k_2 = 5$. This indicates that the LLM's performance benefits from few-shot examples generated by the LLM itself.

We compared the system's performance with the conventional approach of fine-tuning transformer-based models. Specifically, we fine-tuned the multilingual `mT5-base` model (Xue et al., 2021). For training and validation, we introduced errors in gold-standard sentences as described in Section 3 and combined erroneous sentences with the corresponding correct sentences. It resulted in a labeled dataset of 20,000 samples. This dataset was then split into training and validation sets using an 80/20 ratio. The model was trained for 10 epochs with a batch size of 8 and a learning rate of 3×10^{-5}. The model performed well on the synthetic evaluation set. It is likely due to the method used for introducing errors was the same for the synthetic dataset as well as the model's training data.

Table 4 shows the output of the system. To better understand its behavior, we analyzed the system using the Gujarati ERRANT adaptation. The error-type-wise performance of our system is shown in Figure 5. We observe that the system performed relatively well in the case of inflection and other morphological errors. However, it struggled to interpret gender and number for less commonly used nouns; for example, ભસ્મ (*bhasma* — *ash*) is feminine, but the system takes it as neuter. Additionally, it encountered difficulties with language-specific usages, such as પ્રણામ (*praṇām* — a noun for *greeting*) — which is masculine and is always used as plural in Gujarati. The system performed well in spelling and punctuation errors, except for some tricky cases. For example, પાણી (*pāṇī*) means *water*, while પાણિ (*pāṇi*) means *hand*. The system was unable to correct these errors based on

Input:	સંતાતા સૂર્ય સોનેરી રંગ ક્ષિતિજ ઉપર છાંટી હતી.
	santātā sūrya sonerī raṅga kṣitija upar chāṇṭī hatī.
	Translation: The setting sun was spreading a golden hue over the horizon.
Reference:	સંતાતા સૂર્યે સોનેરી રંગ ક્ષિતિજ ઉપર છાંટ્યો હતો.
	santātā sūrye sonerī raṅga kṣitija upar chāṇṭyo hato.
	Translation: The setting sun had spread a golden hue over the horizon.
	સંતાતા સૂર્યએ સોનેરી રંગ ક્ષિતિજ ઉપર છાંટ્યો હતો.
	santātā sūryae sonerī raṅga kṣitija upar chāṇṭyo hato.
	Translation: The setting sun had spread a golden hue over the horizon.
zeroshot:	સંતાતા સૂર્ય સોનેરી રંગ ક્ષિતિજ ઉપર છાંટતો હતો.
	santātā sūrya sonerī raṅga kṣitija upar chāṇṭato hato.
	Translation: The setting sun was scattering a golden hue over the horizon.
Our system:	સંતાતા સૂર્યે સોનેરી રંગ ક્ષિતિજ ઉપર છાંટ્યો હતો.
	santātā sūrye sonerī raṅga kṣitija upar chāṇṭyo hato.
	Translation: The setting sun had spread a golden hue over the horizon.

Table 4: Example illustrating system correction compared to the baseline. While the LLM produces an incorrect correction, our system successfully corrects the sentence.

context. We observe that the system often leaves even simple word-order errors uncorrected. The next section presents the system's ablation study.

5.3 Ablation Study

Configuration	**P**	**R**	**$F_{0.5}$**
Removing M_2	55.59	40.21	51.64
Removing M_1	51.56	31.97	45.93
Without CoT	53.58	36.60	49.03

Table 5: Ablation study showing the effect of each component.

We conducted an ablation study to assess the effects of the prompting technique and memory modules in our system. Specifically, we evaluated the system under three configurations as shown in Table 5. Removing M_2 decreased $F_{0.5}$ by 2.8 points, and substituting Chain-of-Thought prompting with a simple technique decreased $F_{0.5}$ by 5.4 points. The removal of M_1 decreased $F_{0.5}$ by 8.5 points.

6 Conclusion and Future Work

In this work, we propose a system for Gujarati GEC. Our approach integrates non-parametric long-term and short-term memory modules with LLMs. The system requires embedding models, and they are comparatively easier to obtain even for low-resource languages. The integration of memory modules led to significant improvements. It increased the M^2 score by 11.9 points on the human-annotated dataset and 2.9 points on the synthetic dataset.

The modular architecture of our system allows easy replacement of the embedding model and LLM. This makes it possible to adapt the system to other languages. Such flexibility opens new opportunities for developing GEC systems for other low-resource languages with minimal labeled data. Our ERRANT implementation for Gujarati will help to improve the interpretability of future work on Gujarati GEC. Additionally, we provide a detailed analysis of the system's performance across various error categories. This highlights current challenges and identifies areas for future research in Gujarati GEC. The proposed approach can be adopted for other similar Indian languages also, e.g., Hindi and Marathi. With increased use of the system, the proposed memory-based model can be used to generate high-quality error data, which can further be used to develop supervised models.

7 Limitations

The proposed work has the following limitation: our adaptation of the ERRANT toolkit for Gujarati relies on an existing Part-of-Speech tagger, which may introduce noise in error type classification.

References

Khuyagbaatar Batsuren, Omer Goldman, Salam Khalifa, Nizar Habash, Witold Kieraś, Gábor Bella, Brian Leonard, Garrett Nicolai, Kyle Gorman, Yustinus Ghanggo Ate, Maria Ryskina, Sabrina Mielke, Elena Budianskaya, Charbel El-Khaissi, Tiago Pimentel, Michael Gasser, William Abbott Lane, Mohit Raj, Matt Coler, Jaime Rafael Montoya Samame, Delio Siticonatzi Camaiteri, Esaú Zumaeta Rojas, Didier López Francis, Arturo Oncevay, Juan López Bautista, Gema Celeste Silva Villegas, Lucas Torroba Hennigen, Adam Ek, David Guriel, Peter Dirix, Jean-Philippe Bernardy, Andrey Scherbakov, Aziyana Bayyr-ool, Antonios Anastasopoulos, Roberto Zariquiey, Karina Sheifer, Sofya Ganieva, Hilaria Cruz, Ritván Karahóga, Stella Markantonatou, George Pavlidis, Matvey Plugaryov, Elena Klyachko, Ali Salehi, Candy Angulo, Jatayu Baxi, Andrew Krizhanovsky, Natalia Krizhanovskaya, Elizabeth Salesky, Clara Vania, Sardana Ivanova, Jennifer White, Rowan Hall Maudslay, Josef Valvoda, Ran Zmigrod, Paula Czarnowska, Irene Nikkarinen, Aelita Salchak, Brijesh Bhatt, Christopher Straughn, Zoey Liu, Jonathan North Washington, Yuval Pinter, Duygu Ataman, Marcin Wolinski, Totok Suhardijanto, Anna Yablonskaya, Niklas Stoehr, Hossep Dolatian, Zahroh Nuriah, Shyam Ratan, Francis M. Tyers, Edoardo M. Ponti, Grant Aiton, Aryaman Arora, Richard J. Hatcher, Ritesh Kumar, Jeremiah Young, Daria Rodionova, Anastasia Yemelina, Taras Andrushko, Igor Marchenko, Polina Mashkovtseva, Alexandra Serova, Emily Prud'hommeaux, Maria Nepomniashchaya, Fausto Giunchiglia, Eleanor Chodroff, Mans Hulden, Miikka Silfverberg, Arya D. McCarthy, David Yarowsky, Ryan Cotterell, Reut Tsarfaty, and Ekaterina Vylomova. 2022. UniMorph 4.0: Universal Morphology. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 840–855, Marseille, France. European Language Resources Association.

Jatayu Baxi, Om Soni, and Brijesh Bhatt. 2024. Part of speech and morph category prediction for gujarati. *International Journal of Intelligent Systems and Applications in Engineering*, 12(3):586–599.

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*, 49(3):643–701.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

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.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, Atlanta, Georgia. Association for Computational Linguistics.

Sumanth Doddapaneni, Rahul Aralikatte, Gowtham Ramesh, Shreya Goyal, Mitesh M. Khapra, Anoop Kunchukuttan, and Pratyush Kumar. 2023. Towards leaving no Indic language behind: Building monolingual corpora, benchmark and models for Indic languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12402–12426, Toronto, Canada. Association for Computational Linguistics.

Yaxin Fan, Feng Jiang, Peifeng Li, and Haizhou Li. 2023. Grammargpt: Exploring open-source llms for native chinese grammatical error correction with supervised fine-tuning.

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

Y. Gondaliya, P. Kalariya, B. Panchal, and A. Nayak. 2022. A rule-based grammar and spell checking. *SAMRIDDHI : A Journal of Physical Sciences, Engineering and Technology*, 14(01):48–54.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2018. Near human-level performance in grammatical error correction with hybrid machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 284–290, New Orleans, Louisiana. Association for Computational Linguistics.

Gujarati Wikisource contributors. Books - Gujarati Wikisource. Accessed: July 2025.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From rag to memory: Non-parametric continual learning for large language models.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2014. The AMU system in the CoNLL-2014 shared task: Grammatical error correction by data-intensive and feature-rich statistical machine translation. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 25–33, Baltimore, Maryland. Association for Computational Linguistics.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Divesh Ramesh Kubal and Apurva Shrikant Nagvenkar. 2025. Leveraging multilingual models for robust grammatical error correction across low-resource languages. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pages 505–510, Abu Dhabi, UAE. Association for Computational Linguistics.

John Lee. 2004. A classifier-based approach to preposition and determiner errors. In *Proceedings of the 20th International Conference on Computational Linguistics*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *CoRR*, abs/2005.11401.

Wei Li, Wen Luo, Guangyue Peng, and Houfeng Wang. 2025. Explanation based in-context demonstrations retrieval for multilingual grammatical error correction.

Ahmad Musyafa, Ying Gao, Aiman Solyman, Chaojie Wu, and Siraj Khan. 2022. Automatic correction of indonesian grammatical errors based on transformer. *Applied Sciences*, 12(20).

Jakub Náplava and Milan Straka. 2019. Grammatical error correction in low-resource scenarios. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 346–356, Hong Kong, China. 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.

Brijeshkumar Y. Panchal and Apurva Shah. 2024. Spell checker using norvig algorithm for gujarati language. In *Smart Data Intelligence*, pages 281–290, Singapore. Springer Nature Singapore.

Himadri Patel, Bankim Patel, and Kalpesh Lad. 2021. Jodani: A spell checking and suggesting tool for gujarati language. In *2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pages 94–99.

Hitesh Patel and Neelam Patel. 2015. Morphological rule set and lexicon of gujarati grammar. In *Proceedings of VNSGU Journal of Science and Technology*, volume 4, pages 122–126.

Ujjwal Sharma and Pushpak Bhattacharyya. 2025. Hi-GEC: Hindi grammar error correction in low resource scenario. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 6063–6075, Abu Dhabi, UAE. Association for Computational Linguistics.

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.

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Nan Wang, and Han Xiao. 2024. jina-embeddings-v3: Multilingual embeddings with task lora.

Bing Wang, Xinnian Liang, Jian Yang, Hui Huang, Shuangzhi Wu, Peihao Wu, Lu Lu, Zejun Ma, and Zhoujun Li. 2025. Scm: Enhancing large language model with self-controlled memory framework.

Yu Wang, Yuelin Wang, Jie Liu, and Zhuo Liu. 2020. A comprehensive survey of grammar error correction. *CoRR*, abs/2005.06600.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. 2025. From human memory to ai memory: A survey on memory mechanisms in the era of llms.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

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

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386, San Diego, California. Association for Computational Linguistics.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2024. A survey on the memory mechanism of large language model based agents.

A Implementation Details

We accessed `GPT-4o-mini` model via the OpenAI API. The `LangChain`[4] framework was used to build prompting pipelines and streamline model inference. For implementing the memory modules M_1 and M_2, we employed vector database powered by `Milvus`[5], accessed via the `Zilliz` cloud service[6].

We fine-tuned the `mT5-base` model in the Google Colab environment using a T4 GPU with 16 GB of RAM.

B Prompt Templates

B.1 Zeroshot Prompt Template

Task: Correct spelling and grammatical errors in the given Gujarati sentence.

Instructions:

* Only fix errors, do not modify correct sentences, or make unnecessary changes.

* Be confident in corrections. If unsure, leave the sentence unchanged.

* Output only the corrected sentence, no explanations or extra text.

Input Sentence: {sentence_to_correct}

B.2 The Final Prompt Template

Task: Correct spelling and grammatical errors in the given Gujarati sentence.

Instructions:

* Only fix errors, do not modify correct sentences, or make unnecessary changes.

* Be confident in corrections. If unsure, leave the sentence unchanged.

* Output only the corrected sentence, no explanations or extra text.

Example: પહેલો વરસ્યો વરસાદ કે રાફડામાંથી પાંખવાળો મકોડા આકાશે ઊડ્યા આખો દિવસ ઊડ્યા, એકાદ રાત પણ ઉડ્યા; બીજે દિવસે તેનો પાંખો જ્યાં ત્યાં રખડતી આવી જોવામાં?

Let’s think step-by-step.

[4] `https://www.langchain.com/`
[5] `https://milvus.io/`
[6] `https://zilliz.com/`

1. 'વરસાદ' is object and should be preceded by 'વરસ્યો' (verb).

2. 'પાંખવાળો' should be replaced by 'પાંખવાળા' as 'મકોડા' is plural of 'મકોડો'.

3. There should be a semicolon (;) after 'આકાશે ઊડ્યા', because the first clause ends and both clauses are not connected with a connector.

4. There will be a dirgha 'ઊ' in 'ઉડ્યા'.

5. 'પાંખો' is plural and feminine, hence 'તેનો' will be replaced by 'તેની'.

6. 'આવી' (verb) should be preceded by 'જોવામાં', which is a verb used as an adjective (called krudant in Gujarati).

7. The overall sentence is affirmative, so the question mark (?) will be removed and a period should be added.

corrected sentence: પહેલો વરસાદ વરસ્યો કે રાફડામાંથી પાંખવાળા મકોડા આકાશે ઊડ્યા; આખો દિવસ ઊડ્યા, એકાદ રાત પણ ઉડ્યા; બીજા દિવસે તેની પાંખો જ્યાં ત્યાં રખડતી જોવામાં આવી.

Some examples for analysis: {data_from_M_2}

Also, refer to these grammatically correct Gujarati sentences to understand the Gujarati grammar better: {data_from_M_1}

Input Sentence: {sentence_to_correct}

C ERRANT for Gujarati

Our adaptation of the ERRANT toolkit extends the original implementation for use with the Gujarati language.[7] The original version relies on the spaCy pipeline for part-of-speech tagging, tokenization, and morphological analysis, which is not available for Gujarati. Therefore, we implement a custom pipeline tailored to Gujarati. We use a simple tokenizer and a rule-based stemmer from the Gujarati NLP Toolkit[8]. For lemmatization, we design a rule-based lemmatizer using the Unimorph dataset as a dictionary, which includes 16,802 inflected forms of verbs, adjectives, and nouns. Additionally, we use the Hunspell Gujarati dictionary[9] to identify spelling errors. Figure 6 illustrates an example of error tagging performed by our ERRANT adaptation.

[7]https://github.com/chrisjbryant/errant

[8]https://github.com/Rutvik-Trivedi/Gujarati-NLP-Toolkit

[9]https://github.com/harshkothari410/gu-hunspell

```
Input:
    હવે તો ઇશ્વર જ માંર્ગ બતાવાશે.

References:
 1. હવે તો ઈશ્વર જ માર્ગ બતાવશે.
    Now only God will show the way.
 2. હવે તો ઇશ્વર દ્વારા જ માર્ગ બતાવાશે.
    Now the way will only be shown by God.

Edit file:
A 4 5|||R:SPELL|||માર્ગ|||REQUIRED|||-NONE-|||1         // Spelling correction: માંર્ગ → માર્ગ
A 5 6|||R:VERB:INFL|||બતાવશે|||REQUIRED|||-NONE-|||1  // Verb inflection: બતાવાશે → બતાવશે
A 3 3|||M:ADP|||દ્વારા|||REQUIRED|||-NONE-|||2           // Add postposition: દ્વારા
A 4 5|||R:SPELL|||માર્ગ|||REQUIRED|||-NONE-|||2         // Spelling correction: માંર્ગ → માર્ગ
```

Figure 6: Example illustrating the error-annotations generated by the ERRANT toolkit for a given input sentence with multiple reference corrections.