Hi-GEC: Hindi Grammar Error Correction in Low Resource Scenario

Ujjwal Sharma, Pushpak Bhattacharyya
Computation for Indian Langauge Technology (CFILT)
Indian Institute of Technology Bombay, Mumbai, India.
(ujjwalsharma, pb)@cse.iitb.ac.in

Abstract

Automated Grammatical Error Correction (GEC) has been extensively researched in Natural Language Processing (NLP), primarily focusing on English and other resource-rich languages. This paper shifts the focus to GEC for a scarcely explored low-resource language, specifically Hindi, which presents unique challenges due to its intricate morphology and complex syntax. To address data resource limitations, this work explores various GEC data generation techniques. Our research introduces a carefully extracted and filtered, high-quality dataset, HiWikiEdits, which includes human-edited 8,137 instances sourced from Wikipedia, encompassing 17 diverse grammatical error types, with annotations performed using the ERRANT toolkit. Furthermore, we investigate Round Trip Translation (RTT) using diverse languages for synthetic Hindi GEC data generation, revealing that leveraging high-resource linguistically distant language for error generation outperforms mid-resource linguistically closer languages. Specifically, using English as a pivot language resulted in a 6.25% improvement in GLEU score compared to using Assamese or Marathi. Finally, we also investigate the neural model-based synthetic error-generation technique and show that it achieves comparable performance to other synthetic data generation methods, even in low-resource settings.

1 Introduction

The field of Grammatical Error Correction (GEC) involves automatically correcting typographical, syntactic, and fluency errors in written text. Starting in the early 2000s, initially relying on manually crafted rules, the interest in GEC grew significantly during the 2010s, resulting in substantial progress and the development of GEC systems. GEC has evolved through several stages: rule-based, statistical, neural, and language model-based approaches.

Today, the majority of efforts in addressing grammatical errors focus on deep learning and statistical methods rather than rule-based ones. These modern techniques treat GEC as a *translation* problem, converting text from an ungrammatical form to a grammatically correct one. However, these approaches require a significant amount of supervised data in the form of "edits" which are pairs of incorrect and correct sentences (Brockett et al., 2006; Ge et al., 2018; Chollampatt and Ng, 2018; Junczys-Dowmunt et al., 2018).

Despite the widespread interest in GEC, research in this field has largely been concentrated on the English language. This is mainly because there are limited or no benchmark GEC datasets available for other low-resource languages like Hindi.

Due to the difficulty and expense associated with obtaining human-annotated GEC data, and the substantial data requirements for training GEC models, several methods for generating artificial data for GEC have been employed (Izumi et al., 2004; Zhao et al., 2019). These techniques involve introducing noise into error-free sentences using rule-based methods, probabilistic approaches, or round-trip translation to generate errors (Lichtarge et al., 2019; Grundkiewicz et al., 2019).

Model-based error generation approaches have also been explored to generate high-quality synthetic datasets (Xie et al., 2018; Stahlberg and Kumar, 2021). However, these methods are limited by the availability of high-quality seed datasets, typically only accessible for high-resource languages like English. Alternatively, approaches such as extracting corrections from online sources like language learning sites, Wikipedia (Faruqui et al., 2018), and GitHub (Hagiwara and Mita, 2020) offer effective means of collecting data for GEC.

This research paper investigates low-resource GEC for the Hindi language by employing and evaluating various GEC data generation techniques.

Our contributions are:

Proceedings of the 31st International Conference on Computational Linguistics, pages 6063–6075
January 19–24, 2025.

1. HiWikiEdits, a carefully extracted and filtered, high-quality new GEC corpus for Hindi consisting of **8137** human edited sentence pairs. The dataset is extracted from Wikipedia and contains a near uniform distribution of **17** different kinds of errors, with annotations performed using the ERRANT toolkit. The dataset, along with all associated scripts used in its creation, is available for access at link[1]. (Refer Section 3.1.1)

2. A systematic evaluation of synthetic Hindi GEC data generation through Round Trip Translation (RTT) using four languages: English, Assamese, Marathi, and Tamil. For Hindi, the findings indicate that using a high-resource language, regardless of its linguistic divergence, significantly enhances error correction effectiveness compared to mid-resource, linguistically similar languages. Specifically, English as a pivot language achieves a 6.25% improvement in the GLEU score over Assamese and Marathi, which are linguistically related to Hindi. (Refer Table 2)

3. A study of neural model-based synthetic error generation method, that aims to generate errors similar to the seed GEC corpus. Our experiments reveal that this approach achieves comparable performance to other synthetic data generation methods, showcasing its robustness and applicability in low-resource settings. (Refer Table 3)

To the best of our knowledge, no prior study has performed a similar analysis on various techniques for generating synthetic Hindi GEC data. Additionally, there is no manually annotated error correction dataset available for Hindi.

2 Related Work

GEC is frequently regarded as closely related to Machine Translation (MT) (Brockett et al., 2006; Chollampatt and Ng, 2018). The primary objective of GEC is to transform an "ungrammatical" sentence into a "grammatical" one. Neural Machine Translation (NMT) has emerged as a prominent technique for GEC due to its capacity to correct erroneous words, phrases, and sentences not present in the training set (Luong et al., 2015). The introduction of the Transformer architecture (Vaswani, 2017) further solidified NMT as a leading approach for GEC. However, these methods require a significant volume of supervised data comprising *"edits"* sets of incorrect and correct sentence pairs. While significant progress has been made in English and other resource-rich languages, resulting in the development of numerous datasets for evaluating state-of-the-art methodologies (Ng et al., 2014; Grundkiewicz and Junczys-Dowmunt, 2014; Faruqui et al., 2018; Dale et al., 2012; Bryant et al., 2019), low-resource languages, particularly Indic languages, have been relatively neglected.

The creation of manually annotated corpora has significantly advanced GEC technology for English and other resource-rich languages such as Russian (Rozovskaya and Roth, 2019). However, the development of these corpora is both time-consuming and resource-intensive, and such resources are often unavailable for low-resource languages, including Indic languages like Hindi.

To address these challenges, generating artificial data for GEC has become increasingly popular (Izumi et al., 2004; Zhao et al., 2019). This includes adding noise to error-free sentences using rule-based and probabilistic methods, such as token swapping or insertion, and generating errors via round-trip translations through a pivot language (Lichtarge et al., 2019).

Another approach for generating artificial data involves extracting corrections from online resources, including language learning platforms, public revision histories such as Wikipedia (Faruqui et al., 2018), and code repositories like GitHub (Hagiwara and Mita, 2020). Although this method can yield extensive and natural datasets, it has limitations: not all revisions address grammatical errors; some merely enhance content semantically or add information.

Model-based approaches have also been extensively explored for generating high-quality synthetic datasets (Xie et al., 2018; Stahlberg and Kumar, 2021). These methods involve training models on high-quality seed datasets to generate erroneous sentences from error-free ones, effectively simulating the process of introducing grammatical errors. However, these approaches rely on the availability of a high-quality seed corpus.

[1] https://github.com/ujjwalsharmaIITB/Hi-GEC

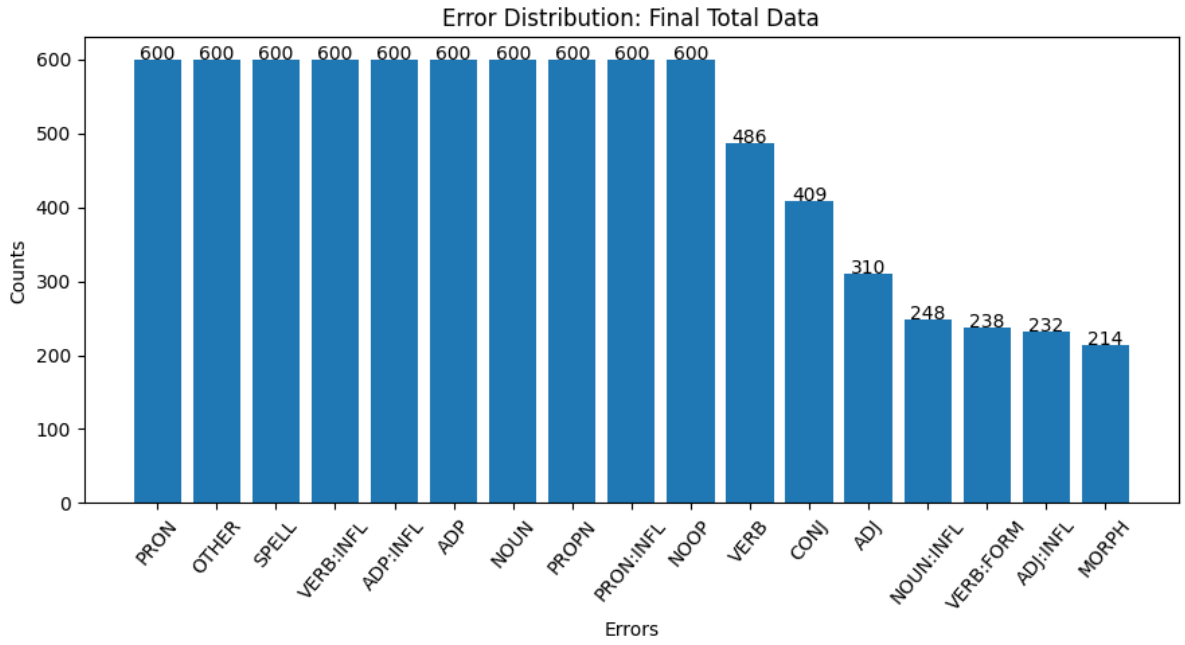

Figure 1: Error Distribution (frequency) of HiWikiEdits.

Dataset Name	# Sentence Pairs	# Tokens
HiWikiEdits	8,137	138,427
Train Split	5,696	96,960
Validation Split	976	16,590
Test Split	1,465	24,877
Direct-Noise	300,000	4,814,365
Round-Trip-Translated	≈ 300,000	5,235,857
Filtered on BLUE (between 30 & 95)	≈ 100,000	-
EGM	≈ 600000	9,938,389

Table 1: Dataset statistics including number of sentences and tokens.

3 Data

3.1 Wikipedia Edits History

Wikipedia provides a data dump of the revision histories of all its pages. This dump includes chronological snapshots of the entire content of each page before and after every edit. Therefore, two consecutive snapshots represent a single revision to the page.

To extract edits from Wikipedia we use the Wikiedits 2.0[2] software which was modified by Sonawane et al. (2020)[3]. We further modified the tool by adding improved filtering and extracted edits from Wikipedia revision dump dated 4 July 2024[4].

We filter the edits using the following constraints:

- We constrained the length of the extracted sentences to be at least 10 words and at most 30 words.
- We constrained the edits such that they must have a difference of no more than 3 words and a Levenshtein edit ratio of less than 0.3.
- We follow Sonawane et al. (2020) and discard the edits containing only a difference in punctuation or numbers and corrections involving rare tokens or HTML markups. We also filtered edits that were related to vandalism and also discarded all the identical pairs.

[2] https://github.com/snukky/wikiedits
[3] https://github.com/s-ankur/wikiedits
[4] https://dumps.wikimedia.org/hiwiki/20240701/hiwiki-20240701-pages-meta-history.xml.7z

3.1.1 HiWikiEdits

After applying the filtering mentioned in section 3.1, we end up with a relatively small corpus of errors, ***HiWikiEdits_full***, which we analyze through the ERRANT[5][6] toolkit. Similar to Sonawane et al. (2020), we found that the error distribution for this corpus is skewed towards certain error types namely, inflectional errors on verbs and adpositions, and some kinds of errors are not grammatical. So we further filter the corpus to contain a limited number of error types and ensure a uniform distribution of these errors. We call this final filtered dataset as ***HiWikiEdits***.

3.1.2 Errors in HiWikiEdits

The HiWikiEdits dataset encompasses a total of 17 distinct error types, as illustrated in Figure 1. These errors have been extracted from an updated version of the ERRANT toolkit. For a detailed explanation of how these errors are obtained, please refer to the official ERRANT repository or to Sonawane et al. (2020).

Furthermore, we have introduced a new category, denoted as *NOOP*, which includes error-free sentences. This addition aims to evaluate the capability of models to accurately identify and process grammatically correct sentences.

3.2 Direct-Noise

The structure of a text can be easily disrupted by making changes such as removing a word or rearranging two words. With the abundance of text available online in different languages, creating significant amounts of artificial training data is relatively easy (Izumi et al., 2004; Grundkiewicz et al., 2019; Lichtarge et al., 2019). Although these types of corruption methods may not always replicate realistic errors made by human writers, they are still beneficial for pre-training GEC models.

To create noisy input sentences, we are using the approach employed by Grundkiewicz et al. (2019). We start with a clean sentence and sample a probability p_{errors} from a normal distribution with a predefined mean and standard deviation. We then multiply p_{errors} by the number of words in the sentence to obtain the number of errors in that particular sentence. After determining the words to modify, we perform one of the following operations for each chosen word, based on a predefined probability: replace the word with one of its ASpell[7] proposals, insert a random word from the dictionary after the current word, delete the word, or swap it with its right-adjacent neighbor. To make the system more robust to spelling errors, the same operations are also used on individual characters with different probabilities. We also add some errors in the *matras*[8] by randomly replacing the current matra with another from a dictionary. In our experiments, we have chosen the mean and standard deviation of the normal distribution to be 0.2 and 0.1, respectively. We are using the following probability values for different types of errors: 0.3 for substitution, 0.15 for insertion, 0.15 for deletion, 0.1 for swapping, and 0.3 for word errors. Additionally, for word errors, we are using the following probability values per character: 0.01 for dropping, 0.06 for swapping, 0.06 for insertion, and 0.06 for matra errors.

3.3 Round Trip Translation

An alternative way to add noise to an error-free sentence is to create a round-trip translation through a pivot language. A pivot language is employed to first convert the original sentence into the pivot language and then back into the original language (Lichtarge et al., 2019). Round-trip translations introduce noise due to weaknesses in the translation models and inherent ambiguities in translation.

In our experiments, we take sentences from the examples that were filtered out during the edits extraction process and create a separate parallel corpus by adding noise to those sentences using round-trip translation. The original sentence from the corpus is the target sentence, and the output of the round-trip translation is the corresponding source sentence. We use a total of 4 different languages from different language families namely, Assamese (Indo-Aryan), English (Indo-European), Marathi (Indo-Aryan), and Tamil (Dravidian). We used these languages because we wanted to assess the effect of using linguistically close and distant languages to Hindi on error correction.

For each pivot language, we create a corrupted dataset by first translating the Hindi sentences into the pivot language and then translating them back to Hindi. We used IndicTrans2 (Gala et al., 2023) for obtaining the translations for all these languages. We also filtered the obtained parallel cor-

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

[6] https://github.com/s-ankur/errant

[7] http://aspell.net

[8] A matra is used to indicate that a vowel is attached to a consonant. (2002)

pus by only taking the sentences that had a BLEU score between 30 and 95. These scores were found empirically by manual inspection.

An overview of dataset statistics is given in Table 1. Figure 2 illustrates an example of the various types of errors introduced by different error-introducing schemes.

4 Error Generation Module

Another method for generating parallel edits involves training an Error Generation Module (EGM). This technique is inspired by backtranslation (Sennrich et al., 2016). The approach entails training a neural sequence-to-sequence model in the reverse direction (**clean** $\rightarrow$ **noisy**) to introduce errors into a clean sentence (Xie et al., 2018). Following the methodology outlined by Xie et al. (2018), we implement a Neural EGM using the Transformer architecture (Vaswani, 2017). By training this end-to-end EGM on a large corpus, the model generates a diverse range of errors that more accurately reflect the noise distribution encountered in real-world data.

In our experiment, we use the training split of *HiWikiEdits* as the seed corpus, augmented with an additional corpus of the Direct-Noise dataset and the best round-trip translated dataset. We validate the EGM on the validation split, which involves generating errors in the opposite direction, to ensure that our model remains consistent with the error distribution present in *HiWikiEdits*.

5 System Overview

To address both error correction and error generation, we employ a neural sequence-to-sequence model built upon the Transformer architecture (Vaswani, 2017).

5.1 Error Correction Module

The Error Correction Module (ECM) is designed to rectify grammatical errors in text. This problem is framed as a machine translation task, where the goal is to translate an ungrammatical sentence into its grammatical counterpart. Consequently, Grammar Error Correction (GEC) utilizes the Encoder-Decoder architecture (Vaswani, 2017).

In this framework, the encoder processes the ungrammatical sentence and produces a latent representation. The decoder then uses this representation to autoregressively generate the grammatical sentence. This process is mathematically represented by Equation 1:

$$P(y|\hat{y}) = \prod_{i=1}^{n} P(y_i|y_{i-1}, \ldots, y_1, \tilde{y}) \quad (1)$$

where y denotes the output (grammatical) sentence, y_i represents the word generated at the i-th step, $\hat{y}$ is the source (ungrammatical) sentence, and $\tilde{y}$ is the encoded representation of $\hat{y}$.

Given a dataset D_E of erroneous sentences, the objective is to maximize the likelihood of the data, typically achieved through cross-entropy loss.

5.1.1 Implementation and Training

All the ECMs are based on the Transformer model (Vaswani, 2017). For detailed information regarding the training procedures, model configurations, and implementation specifics, please refer to Appendix A.

The following approaches were used to train different models :

Base: Training was performed on the train split of the HiWikiEdits dataset.

Direct-Noise: Training was conducted on data generated using the Direct-Noise method.

Base + Direct-Noise : Training was performed using a combination of the training split from the HiWikiEdits dataset and the Direct-Noise dataset in a 6:4 ratio, respectively.

Round Trip Translated (RTT): A total of 10 models were trained using round-trip translated data, including: One model for each pivot language using the complete data. One model for each language pair using filtered data with BLEU scores between 30 and 95. Two models combining all four pivot languages.

Base + Direct-Noise + RTT (best): Based on our experiments, we selected the RTT dataset that achieved the highest GLEU score on the test set. The model was then trained using a combination of the HiWikiEdits training split, the Direct-Noise dataset, and the best-performing RTT data, with a ratio of 6:2:2.

Base + EGM: Finally, a model was trained on the HiWikiEdits training split and the dataset generated by EGM (5.2) in a 6:4 ratio.

5.2 Error Generation Module

EGM is tasked with artificially introducing errors into sentences. This module formulates the problem as a sequence-to-sequence transduction task, where the goal is to transform correct sentences

Source: जिसके नीचे चाँदी में लिखा रहता है " भारत रत्न " , और यह सफ़ेद फीते के साथ गले में पहना जाता है ।   Jisake nīche chāandī main likhā rahatā hai " bhārat ratna " , aur yah safed fīte ke sāth gale main pahanā jātā hai ।
HiWikiEdits: जिसके नीचे चाँदी में लिखा रहता है " भारत रत्न " , और यह सफ़ेद फीते के साथ गले पर पहना जाता है ।   Jisake nīche chāandī main likhā rahatā hai " bhārat ratna " , aur yah safed fīte ke sāth gale par pahanā jātā hai ।
Direct-Noise: जिसके समा नीचे चाँदी में लिखा रहता है " भारत रत्न ", और यह सफ़ेद फीते के साथ गले ◌ं पहना जाता है ।   Jisake samā nīche chāandī main likhā rahatā hai " bhārat ratna ", aur yah safed fīte ke sāth gale ◌ं pahanā jātā hai ।
RTT-Assamese: इसके नीचे चांदी के रंग में "भारत रत्न" लिखा हुआ है, और इसे गले पर सफेद रिबन के साथ पहना जाता है।   Iske nīche chāndī ke ranga main "bhārat ratna" likhā huā hai, aur ise gale par safed riban ke sāth pahanā jātā hai।
RTT-English: जिसके नीचे चांदी में "भारत रत्न" लिखा हुआ है, और इसे सफेद फीता के साथ गर्दन के चारों ओर पहना जाता है।   Jisake nīche chāndī main "bhārat ratna" likhā huā hai, aur ise safed fītā ke sāth gardan ke chāroan or pahanā jātā hai।
RTT-Marathi: जिसके नीचे चांदी में 'भारत रत्न' लिखा हुआ है और इसे गले में सफेद रिबन के साथ पहना जाता है।   Jisake nīche chāndī main 'bhārat ratna' likhā huā hai aur ise gale main safed riban ke sāth pahanā jātā hai।
RTT-Tamil: इसके नीचे चांदी में "भारत रत्न" लिखा हुआ है, जिसे गले में सफेद फीता के साथ पहना जाता है।   Isake nīche chāndī main "bhārat ratna" likhā huā hai, jise gale main safed fītā ke sāth pahanā jātā hai।
EGM: जिसके नीचे चांदी में लिखा रहता है " भारत रत्न " , और यह सफ़ेद फीते के साथ गले में पहना जाता है ।   Jisake nīche chāndī main likhā rahatā hai " bhārat ratna " , aur yah safa◌ेd fīte ke sāth gale main pahanā jātā hai ।

Figure 2: Comparison of Different Error-Generation Schemes. The figure illustrates variations in errors across different methods, including Direct-Noise, RTT-Assamese, RTT-English, RTT-Marathi, RTT-Tamil, and EGM (Error Generation Module). Key differences include discrepancies in phrasing, the use of terms like '*gardan*' vs. '*gale*', '*jiske*' vs. '*iske*' (transliterated in English), the use of prepositions, and minor formatting errors across the different error-introducing schemes. The transliterated versions of the sentences are provided below each example.

into erroneous ones. This approach is analogous to the formulation used for error correction, as described in Equation 1, but in reverse.

5.2.1 Training

Training the EGM follows a similar architecture as ECM and the details can be found in Appendix A. The training involves teaching the model to generate errors that reflect the distribution and types of errors seen in real-world data.

5.2.2 Implementation Details

For the training of the EGM, we use a combination of datasets:

Direct-Noise + RTT (best) + HiWikiEdits Train Split: A model is trained on a dataset combining Direct Noise data, RTT (best) data, and the HiWikiEdits training split in a 2:2:6 ratio, respectively. This combined dataset provides a diverse range of errors and sentence structures to enhance the model's ability to generate realistic and varied errors.

During validation, we ensure that the error generation model produces errors that match the distribution of errors in the validation set. This involves validating the model by generating errors and comparing them with the HiWikiEdits validation set in the opposite direction, thereby ensuring that the model's error generation aligns with the expected error patterns.

6 Results

To evaluate our models, we use the GLEU metric (Napoles et al., 2015) along with the $F_{0.5}$ metric, which is computed using the MaxMatch (M^2) scorer[9] (Dahlmeier and Ng, 2012). Given that the GLEU score demonstrates a stronger correlation with human judgment, it will serve as our primary evaluation metric.

6.1 Quantitative Results

We use the *Base* model as the baseline for comparison. As indicated in Table 3, training with *Direct-Noise* augmented data improved the GLEU score by 11.23. In this zero-shot scenario, where the model is trained exclusively on synthetic data, it outperforms the baseline, which was trained on a relatively small dataset. This improvement suggests that the model benefits from the inclusion of

[9] http://github.com/nusnlp/m2scorer

Round Trip Translated Dataset	English	Marathi	Assamese	Tamil	Combined
Full Data	**44.0**	41.32	41.41	38.25	39.74
Filtered Data	41.54	40.49	37.00	39.20	43.75

Table 2: GLEU scores of models trained on round-trip translated datasets, evaluated on the HiWikiEdits test split.

Dataset used for training	GLEU	Precision (P)	Recall (R)	$F_{0.5}$
Base	55.08	0.098	0.139	0.104
Direct-Noise	66.31	0.011	0.005	0.009
Base + Direct-Noise	71.59	0.213	0.058	0.138
Base + Direct-Noise + RTT (English): M1	73.17*	0.316	0.102	0.223
+ Finetune Train Split: M3	**73.86***	0.403	0.121	0.275
Base + EGM: M2	72.62	0.378	0.059	0.182

Table 3: The results of various models evaluated on the HiWikiEdits test split, including GLEU, Precision (P), Recall (R), and $F_{0.5}$ scores, are summarized. An asterisk (*) indicates improvements are insignificant with respect to M2.

synthetic errors, which enhances its error correction capabilities.

Combining the *Direct-Noise* data with the HiWikiEdits training set further boosts performance, improving the GLEU score by 16.51. The synthetic data helps the model better understand the grammatical structure of the language, and when this knowledge is combined with the train split, it performs well on the unseen test set.

The subsequent experiments investigate cross-lingual transfer in grammar error correction. As outlined in Section 3.3, we employed 10 parallel corpora (4*2 + 2) to train models exclusively on these synthetic datasets and evaluated them on an unseen test set. Table 2 indicates that the model utilizing English as the pivot RTT language achieves a 6.25% higher GLEU score in both filtered and unfiltered scenarios compared to its two next-best counterparts. This improvement is likely due to the fact that the translation model, IndicTrans2 (Gala et al., 2023), was trained on a substantial amount of English-Hindi and Hindi-English parallel corpora compared to the Hi-X (Indic) and X-Hi (Indic) corpora. Additionally, we observe that model performance is better in the filtered combined dataset compared to the unfiltered combined dataset. This finding suggests that the quality of data has a more significant impact on model performance than the quantity of data.

After identifying the most effective RTT dataset, we trained a model (M2) by combining these datasets obtaining an improvement in GLEU score by 18.09 with respect to the base model.

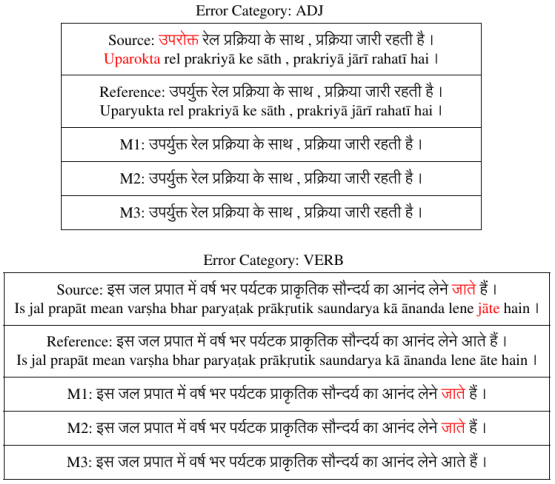

Figure 3: Qualitative analysis of the three best-performing models— M1, M2, and M3 (refer Table 3), focusing on two distinct error types: adjectives and verbs. In the adjective error example ('uparokta' (above) vs. 'uparyukta' (above-mentioned)), all models corrected the error. In the verb error example ('jāte' (going) vs. 'āte' (coming)), M1 and M2 missed the error, while M3 successfully identified and corrected it using context. The transliterated versions of the sentences are provided below the examples.

In our final experiment, we train the ECM using data generated by the EGM, as detailed in Section 5.2. We utilize the same datasets—namely, the target side of the Direct-Noise and RTT corpora—and generate the dataset using beam search with a beam width of 5. As detailed in Table 3, the resulting

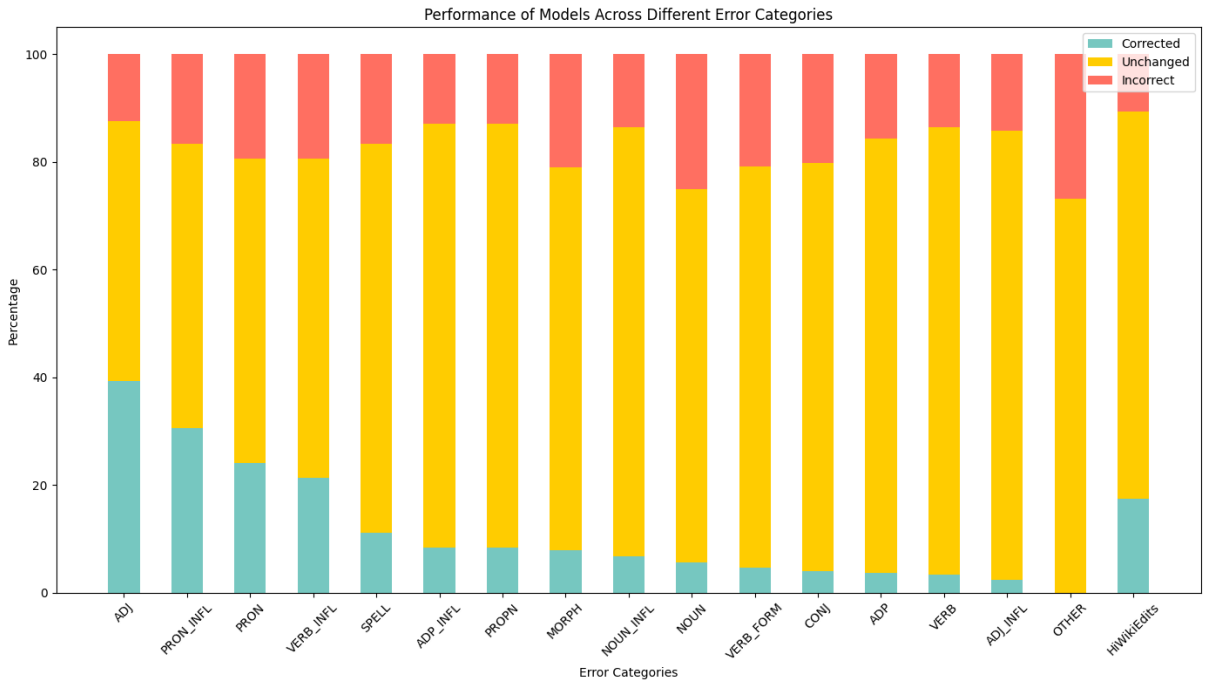

Figure 4: Bar charts illustrating the performance of the error correction module (M3) across different categories, showing the proportions of sentences that were corrected, left unchanged, or incorrectly addressed.

model achieved an improvement of 17.54 GLEU score.

The significance test reveals no significant differences with respect to M1. This result supports the claim that EGMs are beneficial even in low-resource scenarios.

Taking the best model (M2) and fine-tuning it further on the train split of HiWikiEdits resulted in additional improvements of 0.69 GLEU score (M3).

Figure 3 presents sample outputs from our top three models, illustrating their performance on two distinct types of errors. In the first example, which involves an adjective error ('*uparokta*' (above) vs. '*uparyukta*' (above-mentioned)), all three models successfully identified and corrected the error. In the second example, which features a verb error ('*jāte*' (going) vs. '*āte*' (coming)), M1 and M2 failed to detect the error and produced the same output as the source sentence. However, M3 accurately identified the error and corrected it by considering the context of the sentence.

6.2 Error Analysis

The analysis is performed on the best model (M3) to better understand its performance. Figure 4 illustrates that the model performs well in correcting specific types of errors, particularly adjective and pronoun inflections. Additionally, the figure indicates that the model employs a conservative correction strategy. Despite being trained on extensive synthetic data (relative to the HiWikiEdits training split), the model only addresses essential errors and avoids making unnecessary changes to the input, indicating a deliberate and cautious approach to adjustments. For a detailed error analysis, please refer to Appendix C.

7 Conclusion and Future Work

In this paper, we tackle the challenge of grammatical error correction (GEC) for Hindi, a low-resource language with complex linguistic features. We introduced HiWikiEdits, a comprehensive corpus of 8,137 human-edited sentences extracted from Wikipedia, encompassing 17 distinct grammatical error types, with annotations performed using the ERRANT toolkit.

We performed an extensive evaluation of synthetic data generation methods. Specifically, we found that Round Trip Translation (RTT) with a high-resource language, notably English, provides superior error correction results compared to languages that are linguistically closer to Hindi. This finding suggests that linguistic distance may not be as crucial as resource availability and translation model quality in improving GEC performance. Our investigation into the neural model-based synthetic

error-generation technique reveals that it delivers performance comparable to other synthetic data generation methods, even in low-resource scenarios.

Combining various datasets, including DirectNoise and round-trip translated data, improves model performance, with our best model achieving a GLEU score of 73.86. We will release all the synthetically generated Hindi GEC data under the CC-BY-SA 4.0 license publicly for further research.

In the future, we aim to extend the HiWikiEdits dataset by incorporating a broader range of errors. Additionally, we plan to investigate the transfer learning capabilities of our GEC model with linguistically similar languages, such as Marathi.

Overall, our research contributes valuable resources and insights for GEC Hindi, offering methodologies that can be applied to similar languages and advancing the field of grammatical error correction.

8 Limitations

Although the HiWikiEdits corpus is a valuable resource, it is limited to errors found in Wikipedia. This may not fully represent the wide range of grammatical errors encountered in other contexts or types of text. Another limitation is related to the translation models used for round-trip translation (RTT). The uneven training of the model, driven by disparities in the amount of training data, may lead to biases favoring high-resource languages. As a result, the observations regarding cross-lingual transfer may not remain valid if the translation models are trained with an equal or comparable amount of data.

9 Ethics Statement

We have utilized the data extracted from Wikipedia to train and evaluate our models. We acknowledge several key ethical considerations associated with the use of such data. We have taken steps to ensure that any data used is aggregated and anonymized to mitigate any potential privacy concerns. Individual contributions were not identified or singled out in our analyses. Wikipedia, as a crowd-sourced platform, may reflect various biases inherent in its contributors and the sources they cite. We acknowledge the potential for such biases to influence the models trained on this data.

References

Triangle South Asia Consortium 2002. A door into hindi: Devanagari writing system. [Accessed 09-12-2024].

Chris Brockett, William B. Dolan, and Michael Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 249–256, Sydney, Australia. Association for Computational Linguistics.

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.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. *Preprint*, arXiv:1801.08831.

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.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 54–62, Montréal, Canada. Association for Computational Linguistics.

Manaal Faruqui, Ellie Pavlick, Ian Tenney, and Dipanjan Das. 2018. WikiAtomicEdits: A multilingual corpus of Wikipedia edits for modeling language and discourse. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 305–315, Brussels, Belgium. Association for Computational Linguistics.

Jay Gala, Pranjal A. Chitale, Raghavan AK, Varun Gumma, Sumanth Doddapaneni, Aswanth Kumar, Janki Nawale, Anupama Sujatha, Ratish Puduppully, Vivek Raghavan, Pratyush Kumar, Mitesh M. Khapra, Raj Dabre, and Anoop Kunchukuttan. 2023. Indictrans2: Towards high-quality and accessible machine translation models for all 22 scheduled indian languages. *Preprint*, arXiv:2305.16307.

Tao Ge, Furu Wei, and Ming Zhou. 2018. Reaching human-level performance in automatic grammatical error correction: An empirical study. *Preprint*, arXiv:1807.01270.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2014. The wiked error corpus: A corpus of corrective wikipedia edits and its application to grammatical error correction. In *Advances in Natural Language Processing*, pages 478–490, Cham. Springer International Publishing.

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, Florence, Italy. Association for Computational Linguistics.

Masato Hagiwara and Masato Mita. 2020. GitHub typo corpus: A large-scale multilingual dataset of misspellings and grammatical errors. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 6761–6768, Marseille, France. European Language Resources Association.

Emi Izumi, Kiyotaka Uchimoto, and Hitoshi Isahara. 2004. The overview of the SST speech corpus of Japanese learner English and evaluation through the experiment on automatic detection of learners' errors. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

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.

Guillaume Klein, Yoon Kim, Yuntian Deng, Vincent Nguyen, Jean Senellart, and Alexander Rush. 2018. OpenNMT: Neural machine translation toolkit. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 177–184, Boston, MA. Association for Machine Translation in the Americas.

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora generation for grammatical error correction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3291–3301, Minneapolis, Minnesota. Association for Computational Linguistics.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China. Association for Computational Linguistics.

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.

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.

Alla Rozovskaya and Dan Roth. 2019. Grammar error correction in morphologically rich languages: The case of Russian. *Transactions of the Association for Computational Linguistics*, 7:1–17.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Ankur Sonawane, Sujeet Kumar Vishwakarma, Bhavana Srivastava, and Anil Kumar Singh. 2020. Generating inflectional errors for grammatical error correction in Hindi. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop*, pages 165–171, Suzhou, China. Association for Computational Linguistics.

Felix Stahlberg and Shankar Kumar. 2021. 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.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Ng, and Dan Jurafsky. 2018. Noising and denoising natural language: Diverse backtranslation for grammar correction. 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 619–628, New Orleans, Louisiana. Association for Computational Linguistics.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 156–165, Minneapolis, Minnesota. Association for Computational Linguistics.

A Implementation Details

We utilized the *OpenNMT*[10] library (Klein et al., 2018) for training all our models.

During training, each model is validated on a validation split of HiWikiEdits and evaluated on an unseen test set of HiWikiEdits. Early stopping is implemented with patience of 5 epochs, and models are validated at the end of each epoch.

Table 4 provides a comprehensive overview of the hyperparameters used for training the models

Hyperparameter	Value
Encoder Layers	6
Decoder Layers	6
Hidden Size	512
Word-Vector Size	512
Multi-Head Attention Heads	8
Optimizer	Adam
Initial Learning Rate	1.0
Early Stopping Patience	5

Table 4: Hyperparameters used for training the models.

Data tokenization is performed using byte-pair encoding (Sennrich et al., 2015) with the `subword-nmt`[11] tool, employing 32,000 merge operations.

B HiWikiEdits Data Splits

In the process of partitioning the HiWikiEdits dataset, we ensured that the distribution of errors was consistent across all three data splits. Specifically, each split maintained the same percentage distribution of error categories as found in the original dataset. This approach guarantees that each subset of the data accurately represents the overall error distribution, thereby allowing for a fair and unbiased evaluation of model performance across different training, validation, and test sets. By preserving this error distribution, we aim to mitigate any potential skewing of results that could arise from uneven error representation, ensuring that the models are trained and evaluated on data that reflects the true variability and challenges present in the full dataset.

C Error Analysis

Table 5 demonstrates that the model excels at correcting specific types of errors, particularly adjective and pronoun inflections. Additionally, the table reveals that the model adopts a conservative approach when making corrections. Despite being trained on large synthetic synthetic data (relative to the train split of HiWikiEdits), the model only addresses essential errors, indicating a deliberate and cautious approach to adjustments.

To better understand the model's behavior, we analyzed the Levenshtein distance between source and target sentences in two scenarios: where corrections were made and where no corrections were made.

For sentences where no corrections were applied, the mean Levenshtein distance was 2.2 with a standard deviation of 1.5. This relatively small mean distance, along with its low standard deviation, suggests that the model generally preserves the original text when it detects minimal errors.

In contrast, for sentences where corrections were made, the mean Levenshtein distance was 3.3 with a standard deviation of 3.7. This larger mean distance suggests that the model identifies and adjusts more substantial errors, resulting in greater modifications to the text. This indicates that the model can effectively distinguish and correct errors when they are more pronounced.

Figure 6 presents example sentences illustrating cases where the model adopted a conservative approach to making edits. In both examples, the edit distance between the source and target sentences is 1, highlighting minimal changes made by the model.

We also compared the performance of different models across various error categories. Figure 7 illustrates the error-wise GLEU scores for each model. Generally, all models achieve higher scores when there are no errors in the source sentence. This indicates that the models perform optimally under ideal conditions, where the input text does

[10] https://opennmt.net/

[11] https://github.com/rsennrich/subword-nmt

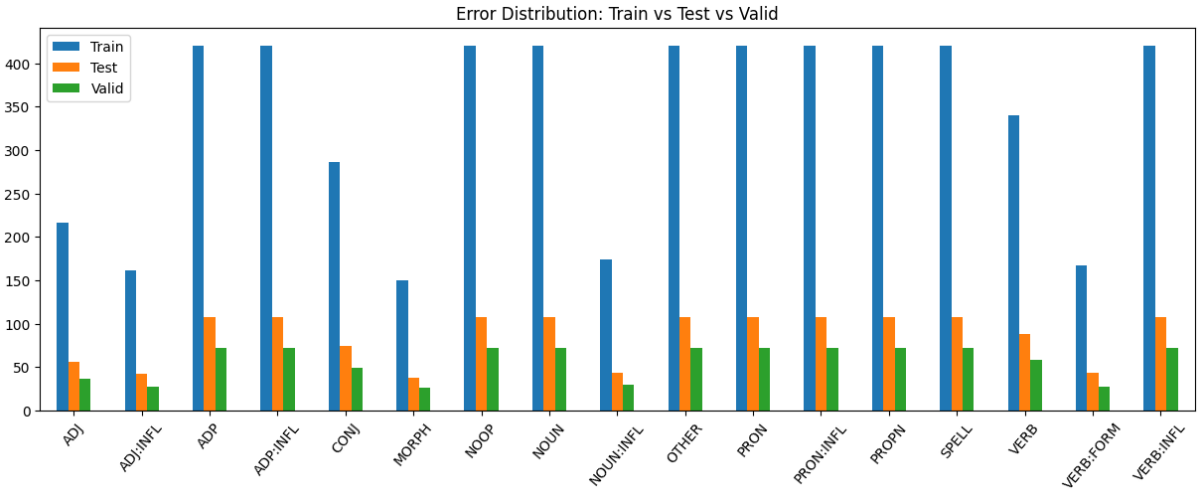

Figure 5: Error Distribution across train, test, and valid splits for HiWikiEdits.

Error Category	Total Sentences	Corrected	Unchanged	Incorrect
NOOP	108	-	97 (89.81%)	11 (10.18%)
ADJ	56	22 (39.29%)	27 (48.21%)	7 (12.5%)
PRON_INFL	108	33 (30.56%)	57 (52.78%)	18 (16.66%)
PRON	108	26 (24.07%)	61 (56.48%)	21 (19.44%)
VERB_INFL	108	23 (21.30%)	64 (59.26%)	21 (19.44%)
SPELL	108	12 (11.11%)	78 (72.22%)	18 (16.66%)
ADP_INFL	108	9 (8.33%)	85 (78.70%)	14 (12.96%)
PROPN	108	9 (8.33%)	85 (78.70%)	14 (12.96%)
MORPH	38	3 (7.89%)	27 (71.05%)	8 (21.05%)
NOUN_INFL	44	3 (6.82%)	35 (79.55%)	6 (13.63)
NOUN	108	6 (5.56%)	75 (69.44%)	27 (25%)
VERB_FORM	43	2 (4.65%)	32 (74.42%)	9 (20.93%)
CONJ	74	3 (4.05%)	56 (75.68%)	15 (20.27%)
ADP	108	4 (3.70%)	87 (80.56%)	17 ((15.74)
VERB	88	3 (3.41%)	73 (82.95%)	12 (13.63)
ADJ_INFL	42	1 (2.38%)	35 (83.33%)	6 (14.28%)
OTHER	108	0 (0.00%)	79 (73.15%)	29 (26.85)
HiWikiEdits	1465	256 (17.47%)	1053 (71.88%)	156 (10.64%)

Table 5: Performance of Models Across Different Error Categories Sorted by Corrected Percentage.

Source: इस रणनीति को भारत ने पूर्वी पाकिस्तान (आज का बंग्लादेश) मे भारत पाकिस्तान के बीच तीसरा युद्ध मे सफलता पूर्वक प्रयोग किया । Is raṇanīti ko bhārat ne pūrvī pākistān (āj kā banglādesh) me bhārat pākistān ke bīch tīsarā yuddha me safalatā pūrvak prayog kiyā ।
Target: इस रणनीति को भारत ने पूर्वी पाकिस्तान (आज का बंग्लादेश) मे भारत पाकिस्तान के बीच तीसरे युद्ध मे सफलता पूर्वक प्रयोग किया । Is raṇanīti ko bhārat ne pūrvī pākistān (āj kā banglādesh) me bhārat pākistān ke bīch tīsare yuddha me safalatā pūrvak prayog kiyā ।
Source: उनके अनेक कविताओं में से सबसे सर्वश्रेष्ठ कविता हैं कोंकणचे शेतकार। Unake anek kavitāoan mean se sabase sarvashreṣhṭha kavitā hain koankaṇache shetakāra।
Target: उनके अनेक कविताओं में से सबसे सर्वश्रेष्ठ कविता है कोंकणचे शेतकार। Unake anek kavitāoan mean se sabase sarvashreṣhṭha kavitā hai koankaṇache shetakāra।

Figure 6: Sentences where the model produced the same output, the edit distance of both of these sentences is 1.

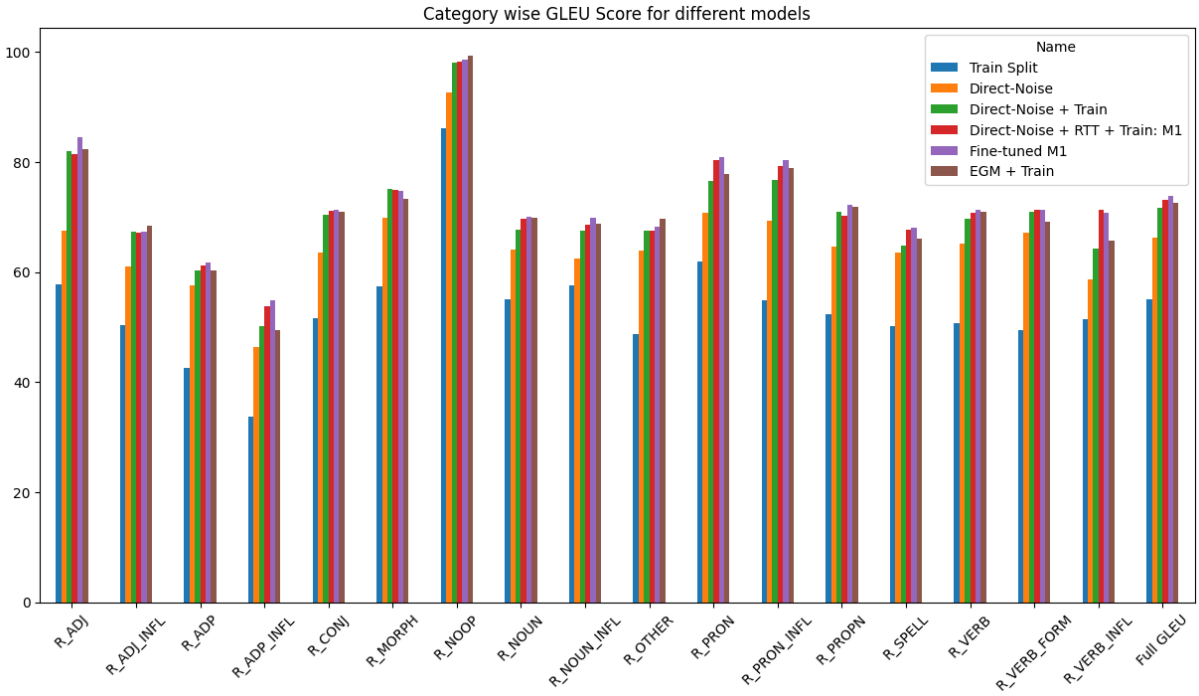

Figure 7: Error-wise GLEU scores for different models across various error categories. The figure shows that models perform better when no errors are present in the source sentences. Furthermore, performance improves with larger and higher-quality training datasets.

not contain any mistakes or imperfections.

The results in Figure 7 also reveal a trend where the performance of the models improves as the volume and quality of the training data increase. More comprehensive and diverse training datasets seem to enhance the models' ability to handle errors and produce higher GLEU scores, suggesting that a larger and better-curated training corpus contributes to better model robustness and accuracy.