Urdu Grammar Error Correction

Prepared For:
Muhammad Adeel
Lecturer, NLP – A4,
University of Management and Technology

By:

Minaam Ahmad	**F2021266555**
Azka Khalid	**F2021266571**
Abdul Rafay	**F2021266352**

24 June, 2025

Abstract:
Grammar Error Correction is an active area of research within the field of NLP, yet its scope remains restricted to English and other resource-rich languages. Urdu is a language that is widely spoken in South Asia. However, due to the lack of annotated datasets no work has been in field of GEC for Urdu. This project presents an end-to-end neural approach for Urdu Grammar Error Correction using the mT5 transformer model. We develop a synthetic error generation pipeline that creates 10,000 parallel Urdu sentences with seven linguistically-motivated error types: gender disagreement, tense errors, number disagreement, word order mistakes, spelling errors, missing words, and extra words. The dataset captures common morphological and syntactic errors specific to Urdu's gender-inflected verb system and SOV sentence structure. Additionally, our study is the first to focus on GEC systems, as to the best of our knowledge, no prior work has been done in this field.

Table of Contents

Table of Figures

Table of Tables

Introduction:

The goal of this project is to correct the given grammatically incorrect Urdu sentence using Deep Learning Techniques. We used a pre-trained mT5 model developed by google to fine tune it on our dataset created synthetically. Apart from the GEC model, this paper presents an annotated dataset for GEC in Urdu language.

Literature Work:

Although there has been significant progress in the field of GEC for English and other resource rich languages, to the best of our knowledge no has work has been in the field of GEC for Urdu language. Consequently, our literature review focuses on works on GEC for other languages. In addition, we also focus on works that generate synthetic datasets for low resource languages.

In their paper, (Naghshnejad et al., 2020) presented a general survey of the recent Deep Learning based approaches for Grammar Error Handling. Their findings can be in seen Table 1.

Model	**Precision**	**Recall**	**F0.5*F*0.5**
RNN NMT (Zheng & Briscoe, 2016)	65	39.0	30.4
CNN (Cholhapati & Ng, 2018)	65.5	33.1	34.8
RNN-Transformer (Dreyer-Dowmunt, 2018)	66.8	34.5	36.3
Copy-augmented Transformer (Zhao, et al., 2019)	71.6	38.7	61.2
PIE (Awasthin, et al., 2019)	68.3	43.2	61.2

Table 1: Table reproduced from (Naghshnejad et al., 2020)

In their paper, (Solyman et al., 2019) proposed a Deep Learning based GEC model for the Arabic language. The authors introduced an encoder-decoder model utilizing multiple convolutional layers and an attention mechanism. They tested the proposed model on the Qatar Arabic Language Bank (QALB) test corpus. Precision, recall, and F1 score were used as evaluation criteria. The model achieved a precision score of 70.23%, a recall score of 72.10%, and an F1 score of 71.14%.

After focusing on GEC systems for different languages, we will now focus on different strategies to create synthetic GEC datasets.

The deliberate injection of errors into grammatically correct sentences has emerged as a critical strategy for overcoming the limited availability of training data. Errors can injected by using a variety of approaches including rule-based systems and round-trip translation (Izumi et al., 2004; Budi Irmavati, 2017; Foster and Andersen, 2009). The limitation of deliberate injection of grammatical errors is that artificial errors should mirror actual errors closely in order to create a dataset is reliable for training and reflective of real-world language use.

Another strategy that is commonly used is the extraction of edit histories from the websites that maintain public revision histories such as Wikipedia. Synthetic dataset generated using this strategy mimics real world dataset because the edits represent actual grammatical mistakes made by humans. However, since these edits also contains other than grammatical mistakes, they need to be filtered. Consequently, make this process challenging (Grundkiewicz and Junczys-Dowmunt, 2014; Boyd, 2018; Faruqui et al., 2018).

In their paper, (Sonawane et al., 2020) combine the two strategies mentioned above to generate a synthetic dataset containing inflectional errors for Hindi language. In addition, they train a base Transformer model and two state of the art English GEC model to create a baseline for GEC in Hindi Language. F0.5$F0.5$ and GLEU score are used as an evaluation criteria. The training dataset is created by using a rule-based framework whereas the test dataset is filtering Wikipedia Edit History in Hindi using ERRANT. Since Hindi is similar to Urdu, we follow similar approach as taken by (Sonawane et al., 2020) to develop a GEC model for Urdu.

Requirements:

Dataset: Facial images annotated with specific measurements.
Libraries and Tools

- Python
- Google Colab
- mT5-small
- Transformers, pyTorch, numpy, sklearn

Data Collection

About 10,000 Urdu sentences were collected from web scrapping, Wikipedia, news etc.

Sample Data:

	A
1	
2	سچن ٹینڈولکر کا وراثت کوہلی کو مشورہ، یہ غلطی کبھی مت کرنا، تباہ ہو جائے گا کیریئر
3	گریٹر نوئیڈا میں ماں بیٹی کا سفاکانہ قتل، معاملہ میں ایف آئی آر درج، تحقیقات شروع
4	عصمت چغتائی کے 107 ویں یوم پیدائش پر گُوگل کا ٹرِبیوٹ کے ذریعہ اردوادبیہ کو خراج عقیدت
5	فلم اسٹار انوپم کھیر فلم اینڈ ٹیلی ویژن انسٹی ٹیوٹ آف انڈیا کے چیئرمین مقرر
6	جموں۔ کشمیر: 9 سالہ بچی کی اجتماعی عصمت دری کے بعد آنکھیں نکال کر جسم پر چھوڑا تیزاب
7	بال بال بچ گئے نمبر ایک گیند باز جسپریت بمراہ ، ورلڈ کپ سے پہلے ختم ہوجاتا کرکٹ کیریئر ! .
8	جنوبی افریقی گیند بازوں کا قہر ، بابراعظم کی نصف سنچری،181رنوں پر آل آوٹ ہوگئی پاکستانی ٹیم
9	فلم ' ٹیگس آف ہندستان ' کے ٹریلر میں کٹرینہ کیف کا نظر آیا یہ بولڈ انداز ، یہاں دیکھیے تصویریں
10	آئی پی ایل 2018 : ممبئی - بنگلورو میچ کے دوران کوہلی کے ساتھ بڑا دھوکہ ، "سازش "میں امپائر بھی شامل
11	پرینکا چوپڑا کے کہنے پر کینسر سے متاثر سونالی بیندرے نے اپنایا نیا انداز، کیا شکریہ
12	سلمان خان کی اس فلم میں یوپی کے بائیکرس دکھائیں گے اپنا کرتب ، دیکھیے کو ملے گا "موت کا کنواں" .
13	سلمان خان نے بہترین اداکاری کے ذریعہ فلمی دنیا میں قائم کی اپنی ایک علاحدہ شناخت
14	پاکستان کوملا 15 سال کا طوفانی گیند باز ، دوسرے ہی میچ میں حاصل کرلئے 6 وکٹ
15	بی جے پی میں شامل ہوں گی سیما چودھری؟ دو- تین دن میں آسکتی ہے بڑی خبر
16	"رنبیر کپور نے کیا قبول ، بے انہیں بھی عالیہ پر "کرش
17	بولی پر اداکارہ کی کار پر شرابیوں کا حملہ ، لوگوں نے نہیں کی مدد تو خود ہی چپل سے کرنے لگی پٹائی
18	جانیں رنبیر کپور پر ان کی کرائے دار نے کیوں کیا 50 لاکھ کا مقدمہ
19	دھونی اینڈ کمپنی کی کولکاتہ نائٹ رائیڈرس پرآسان جیت، رسیل نے اس میچ میں بھی دکھایا دم
20	پرنیتی چوپڑا نے بلیک بکنی پہن کر پانی میں لگائی آگ ، دیکھ کر آپ کے اڑجائیں گے ہوش
21	خوشحالی کیلئے گھر پر بلّی ٹائلرک ، خاتون کی آنروریزی کرکے ہوا فرار
22	پونے : ایک سالہ بچی سے عصمت دری کے بعد کیا قتل، ملزم گرفتار
23	اغوا کے بعد تین سال کی معصوم کی آنروریزی ، قبرستان کے نزدیک خون سے لت پت ملی ، حالت نازک
24	

Figure 1: Sample Data

Data Annotation:

The Sentences were annotated using a python script. We Injected the grammatical errors in the sentences.

The common Urdu grammar error that was injected:

```
error_types = [
    "gender_disagreement",
    "tense_error",
    "number_disagreement",
    "word_order",
    "spelling_error",
    "missing_word",
    "extra_word"
]
```

Incorrect: علی اسکول جاتی ہے	Correct: علی اسکول جاتا ہے	(Gender error – "جاتی" instead of "جاتا")
Incorrect: علی جاتا اسکول ہے	Correct: علی اسکول جاتا ہے	(Word order error – unnatural sequence)
Incorrect: علی شچولے جاتا ہے	Correct: علی اسکول جاتا ہے	(Spelling/phonetic error – "شچولے" instead of "اسکول")

After the errors were injected using python script, all the dataset was formatted in a json file with proper mapping, as a result we get 10,000 pairs of Urdu correct and incorrect sentences making it suitable for Urdu GEC.

Dataset Sample:


```
"pairs": [
  {
    "correct": "استاد نے کمپیوٹر سویا۔",
    "incorrect": "استاد نے کمپیوٹر سویاں۔",
    "error_type": "number_disagreement",
    "error_description": "Singular/plural disagreement",
    "target_verb": "سویا"
  },
  {
    "correct": "کیا بلی نے کتاب کھیلی؟",
    "incorrect": "کےا بلے نے کتاب کھیلے؟",
    "error_type": "spelling_error",
    "error_description": "Common spelling mistakes",
    "target_verb": "کھیلی"
  },
  {
    "correct": "گھوڑی کل گھر پر پڑھتی ہے۔",
    "incorrect": "پر کل گھوڑی گھر پڑھتی ہے۔",
    "error_type": "word_order",
    "error_description": "Incorrect word order",
    "target_verb": "پڑھتی ہے"
  },
  {
    "correct": "صبح وہ لوگ بازار پر پڑھے۔",
    "incorrect": "صبح وہ لوگ بازار پر پڑھیں۔",
```


Figure 2: Sample Dataset

Data Processing:

1.Loading and Parsing JSON Data

- The dataset was loaded from a JSON file named 'urdu_error_dataset.json'.
- The file was expected to contain a dictionary with a key "pairs", each pair being a dictionary with "incorrect" and "correct" sentence fields.

```
with open('urdu_error_dataset.json', 'r', encoding='utf-8') as f:
    data = json.load(f)
pairs = data['pairs']
```

2. Train-Validation Split

- The full dataset of sentence pairs was split into **90% training** and **10% validation** using train_test_split from sklearn.
- A fixed random_state (seed = 42) was used for reproducibility.

```
train_pairs, val_pairs = train_test_split(pairs, test_size=0.1, random_state=seed)
```

3. Tokenization using MT5Tokenizer

- The MT5Tokenizer was used to tokenize both **incorrect (source)** and **correct (target)** sentences.
- Tokenization included:
    - **Padding** to a fixed max length (128 tokens).
    - **Truncation** of sentences exceeding the max length.
    - Conversion to PyTorch tensors.

```
tokenizer(source, max_length=128, padding='max_length', truncation=True, return_tensors='pt')
```

4. Special Token Injection (Optional)

- Special tokens "<correct>" and "<incorrect>" were added to the tokenizer vocabulary, although not used directly in inputs during training (but can be reserved for future improvements like tagged sentence types).

```
special_tokens = ["<correct>", "<incorrect>"]
tokenizer.add_tokens(special_tokens)
```

5. Label Preprocessing (Padding Masking)

- During training, the **padding token IDs** in target sequences (labels) were replaced with -100, which tells PyTorch's loss function to **ignore** these positions when computing loss.

```
labels[labels == tokenizer.pad_token_id] = -100
```

6. Flattening Input Tensors

- Tokenized tensors for each sentence pair were flattened before being passed into the DataLoader, ensuring compatibility with model input.

7. Batching with DataLoader

- Custom PyTorch Dataset class (UrduGECDataset) was created for structured access to training samples.
- Batches were generated using DataLoader with:
    - shuffle=True for training set (randomized batches each epoch)
    - Fixed batch_size=8

8. Device Transfer

- Tensors (input_ids, attention_mask, and labels) were explicitly transferred to **GPU or CPU** using .to(device) before being used in model operations.

Summary Table

Step	**Technique/Tool Used**	**Purpose**
JSON Parsing	json.load()	Load Urdu sentence pairs
Train-Test Split	train_test_split (scikit-learn)	Create training and validation datasets
Tokenization	MT5Tokenizer	Convert text into model input format
Padding & Truncation	padding='max_length', truncation=True	Ensure fixed input size for model
Padding Masking	labels[pad_token_id] = -100	Prevent model from learning on padding
Tokenizer Vocabulary	add_tokens()	Extend tokenizer with special tokens
Batch Generation	DataLoader	Feed model in batches for efficiency
Device Management	tensor.to(device)	GPU acceleration during training

Table 2: Summary of Preprocessing

Model Training:

The model used for this task was google/mt5-small, a multilingual encoder-decoder transformer pretrained on the mC4 dataset. It was fine-tuned specifically for the task of Urdu GEC using a parallel dataset of incorrect–correct sentence pairs.

Training Setup

- **Model**: MT5ForConditionalGeneration (mT5 Small)
- **Tokenizer**: MT5Tokenizer with additional special tokens (<correct>, <incorrect>)
- **Optimizer**: AdamW with a learning rate of 3e-4
- **Scheduler**: Linear warm-up schedule with 100 warm-up steps
- **Batch size**: 8
- **Epochs**: 10
- **Max input/output length**: 128 tokens
- **Loss function**: Cross-entropy loss with padding tokens masked out (-100)

Training was performed using PyTorch, with support for GPU acceleration when available.

Loss Metrics Across Epochs

The model showed **consistent learning progress** and decreasing validation loss across the 10 training epochs, indicating successful fine-tuning. Here's a summary of training and validation loss per epoch:

Epoch	Train Loss	Validation Loss	Model Saved
1	5.5294	1.7834	✅ Yes
2	2.1133	1.2638	✅ Yes
3	1.5655	1.0613	✅ Yes
4	1.2717	0.7380	✅ Yes
5	0.8209	0.2302	✅ Yes
6	0.3686	0.1544	✅ Yes
7	0.2560	0.1176	✅ Yes
8	0.2000	0.1028	✅ Yes
9	0.1671	0.0941	✅ Yes
10	0.1514	0.0878	✅ Yes

Table 3: Training Results

- The **final validation loss** was **0.0878**, showing high performance in predicting correct grammatical sentences from erroneous ones.
- After each epoch, the model was evaluated on the validation set, and the best-performing checkpoint was saved based on validation loss.

Model Evaluation

After training, the best checkpoint (best_model_urdu_gec.pt) was loaded, and the model was tested on unseen Urdu sentences with both errors and correct forms.

Sample Inference Results

The model correctly identified and fixed grammar errors while leaving correct sentences unchanged:

Original: استاد نے کمپیوٹر سویاں۔

Corrected: استاد نے کمپیوٹر سویا۔

Original: کیا بلی نے کتاب کھیلی؟

Corrected: کیا بلی نے کتاب کھیلی؟

Original: گھوڑی کل گھر پر پڑھتی ہے۔

Corrected: گھوڑی کل گھر پر پڑھتی ہے۔

- **First sentence**: Fixed gender/verb mismatch ("سویا" → "سویاں")
- **Second and third sentences**: Left unchanged as they were grammatically correct, showing that the model **does not overcorrect**.

ChrF

The model achieved a **ChrF score of 92.30**, indicating a very high character-level similarity between the predicted corrections and the ground truth. This suggests the model is effectively learning fine-grained grammatical patterns in Urdu and performing well in correcting common errors with high precision and recall.

How ChrF Works (Simplified):

1. **Break both prediction and reference into character-level n-grams**
   For example, using 6-grams (as in our output):
   - Prediction: "گھر پر پڑھتی" → ["ر پر پ", "ھر پر ", "گھر پر", ...]

- Reference: "گھر پر پڑھتا" → ["گھر پر", "ھر پر ", "ر پر پ", ...]

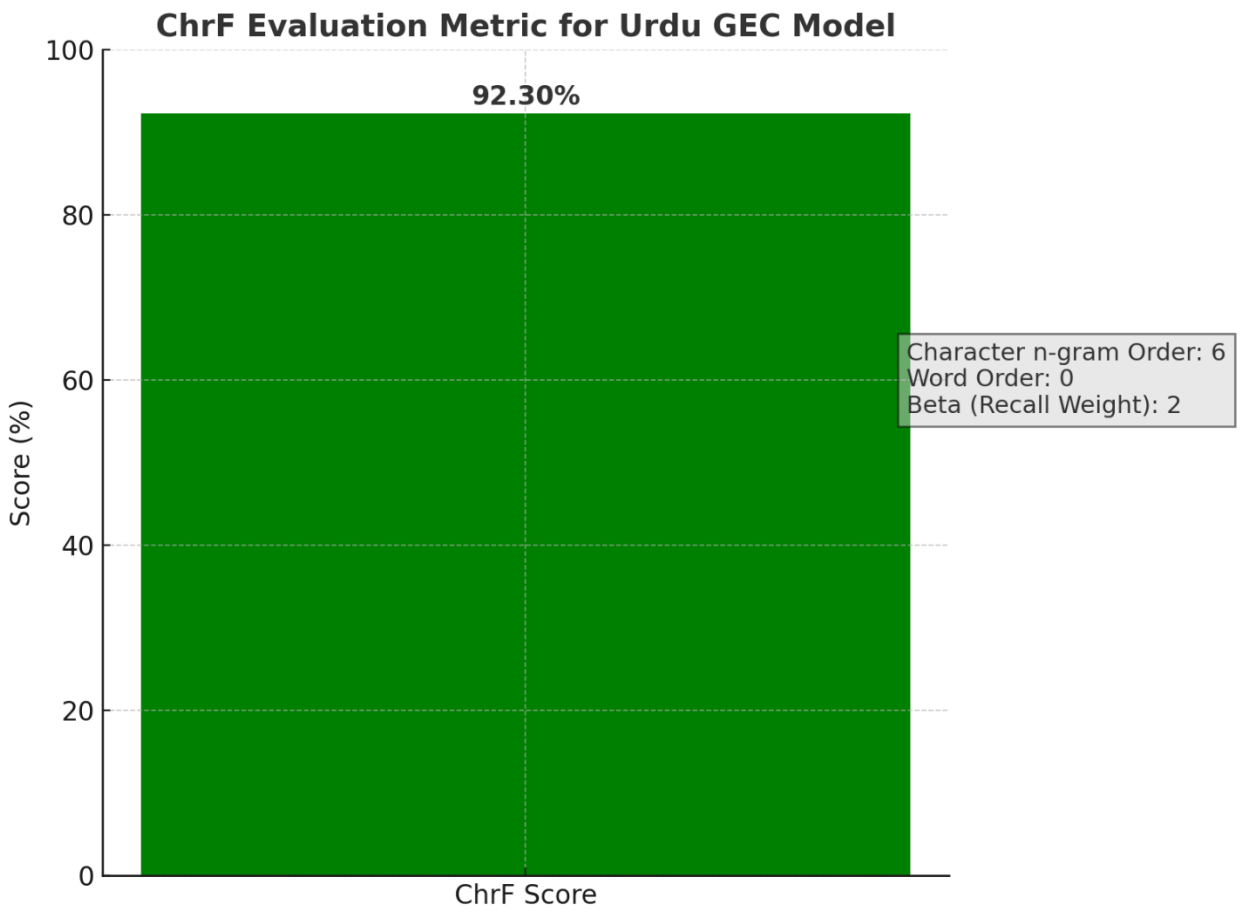

Figure 3: ChrF Evaluation

ChrF: {'score': 92.84, 'char_order': 6, 'word_order': 0, 'beta': 2}

Limitations & Future Work

In this section, we highlight the major limitations of our work.

As stated earlier, there was no existing dataset for Urdu GEC. This meant that we had to curate a dataset using manually. However, creating a dataset manually is a cumbersome process. As a result, we scrapped a dataset from the website and artificial injected grammatical errors. Consequently, our model only deals with substitution INFLECTIONAL errors.

During the data collection phase, we scrapped children's stories from as they contained shorter and simpler sentences, in order to facilitate model training. However, this means that our model can only deal with simple sentences of a moderate length at one time.

As mentioned earlier, our algorithm for artificially injecting errors is designed such that a sentence contains either 1, 2 or 3 grammatical which returns only a single error per sentence. This means that we cannot effectively evaluate our model for sentences that contain multiple errors.

Future work will include creating a more natural and complex error datasets with logical sentences. Due to limitation of hardware support we used free version of google colab, if we had GPU access available we could have used mt5-base and results will be more promising.

Conclusion

Automated Urdu GEC is a complex and very vast task but this project demonstrates and provides the basic setup and working version using a synthetic dataset. However the results were good as per the dataset but if we can improve the dataset and can use the more enhanced version of mT5 which is mT5-base or mT5-large the results will increase significantly.

References

- [1] M. Naghshnejad, P. Moradi, and B. Minaei, “A survey on grammatical error correction,” *arXiv preprint arXiv:2005.11919*, 2020.

- [2] X. Zheng and T. Briscoe, “Automated grammatical error detection for language learners,” in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, 2016, pp. 1072–1082.

- [3] S. Cholhapati and H. T. Ng, “Revisiting neural error detection and correction,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2575–2585.

- [4] M. Junczys-Dowmunt and R. Grundkiewicz, “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*, 2018, pp. 595–606.

- [5] W. Zhao, R. Wang, K. Liu, and J. Zhao, “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*, 2019, pp. 1568–1578.

- [6] A. Awasthi, S. Gupta, D. Singh, and S. J. S. M. J. S. Bhatnagar, “Parallel iterative editing for local sequence transduction,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.

- [7] A. Solyman, M. Elmahdy, and A. Khalaf, “Arabic grammar error correction using neural networks,” in *Procedia Computer Science*, vol. 163, 2019, pp. 325–333.

- [8] E. Izumi, K. Uchimoto, and H. Isahara, “Automatic error detection in the Japanese learner's English spoken data,” in *Proceedings of the 20th International Conference on Computational Linguistics*, 2004.

- [9] B. Irmavati, “Automatic generation of erroneous sentences using syntax-based patterns,” *IJCSMC*, vol. 6, no. 1, pp. 134–141, 2017.

- [10] J. Foster and J. Andersen, “GenERRate: Generating errors for use in grammatical error detection,” in *Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, 2009.

- [11] R. Grundkiewicz and M. Junczys-Dowmunt, “The WikEd error corpus: A corpus of corrective Wikipedia edits and its application to grammatical error correction,” in *Proceedings of LREC*, 2014.

- [12] A. Boyd, “Generating training data for grammatical error correction using Wikipedia revision histories,” *University of Washington*, 2018.

- [13] M. Faruqui, H. Sajjad, R. Grishman, Y. Tsvetkov, and C. Dyer, “WikiSplit: A sentence splitting corpus from Wikipedia edits,” *arXiv preprint arXiv:1808.09468*, 2018.

- [14] R. Sonawane, S. Pawar, A. Saini, and A. Aralikatte, “A grammar error correction corpus and baseline system for Hindi,” in *Proceedings of the 12th Language Resources and Evaluation Conference (LREC)*, 2020.