Define Transliteration

Converting text from one script (here, Roman Hindi) to another script (here, Devanagari) while preserving pronunciation.

Input format:

•A string of Roman Hindi text (either a single word or a sentence).
•Example (word): "`namaste`"
•Example (sentence): "`aap kaise ho`“

Output format:

•A string of Devanagari text corresponding to the input pronunciation.
•Example (word): "नमस्ते"
•Example (sentence): "आप कैसे हो"

Data downloading and cleaning

Converting text from one script (here, Roman Hindi) to another script (here, Devanagari) while preserving pronunciation.

- **How much data did you use?**
  1,064,180 pairs of Hindi Roman → Devanagari.

- **What kind of sampling of the data did you do?**
  None; we used the dataset as-is.

- **From what source?**
  Aksharantar dataset on Hugging Face

- **Did you use any cleaning? If so, what and how, and why?**
  Yes. We removed duplicates where a single Hindi Roman input corresponded to multiple Devanagari outputs. In such cases, we kept only the output with the highest score to ensure a one-to-one mapping. (234975)

```
test Loss: 0.3220 | test ACC: 0.5011 | test F1: 0.8943
```

```
  test Loss: 0.3220 | test ACC: 0.5011 | test F1: 0.8943
    test Loss: 0.3220 | test ACC: 0.5011 | test F1: 0.8943
```

Non ml based model

We employed a **rule-based transliteration model** from the indic_transliteration library to convert Hindi words written in Roman script into Devanagari script. Rule-based models like this typically work by mapping characters or sequences of characters from the source script to corresponding characters in the target script based on predefined linguistic rules. For example, sequences like "aa" might be mapped to "आ", "ch" to "च", and so on. The model follows these deterministic character-level or phonetic rules to generate transliterations.

However, this approach has significant limitations. While it can handle straightforward character mappings, it **fails to capture the broader context of words or sentences**, which is crucial for accurate transliteration, especially in cases of ambiguity or irregular spellings. As a result, the model’s **overall performance is low**, achieving an F1 score of **0.7001**, which indicates moderate overlap with the reference transliterations, but the **word-level accuracy is only 0.0518**, highlighting that very few words were correctly transliterated in their entirety

LSTM Based Transliteration

What did you read for this part of the assignment?

EE782 Lectures for lstm and bilstm models

- Compare greedy decoding vs beam search

  LSTM:

  Test Loss_greedy: 0.3339, Test ACC_greedy: 0.4828, Test F1_greedy: 0.8895

  Test Loss_beam: 0.3339, Test ACC_beam: 0.4924, Test F1_beam: 0.8913

  BILSTM:

  Test_all_Loss_greedy: 0.3214, Test_all_ACC_greedy: 0.4919,Test_all_F1_greedy: 0.8941

  Test_all_Loss_beam: 0.3214, Test_all_ACC_beam: 0.4975, Test_all_F1_beam: 0.8944
- Is your program running?

  yes

```
test Loss: 0.3220 | test ACC: 0.5011 | test F1: 0.8943
  test Loss: 0.3220 | test ACC: 0.5011 | test F1: 0.8943
    test Loss: 0.3220 | test ACC: 0.5011 | test F1: 0.8943
```

Transformer Based Transliteration

What did you read for this part of the assignment?
https://arxiv.org/pdf/2208.10801
https://aclanthology.org/2023.findings-emnlp.4.pdf

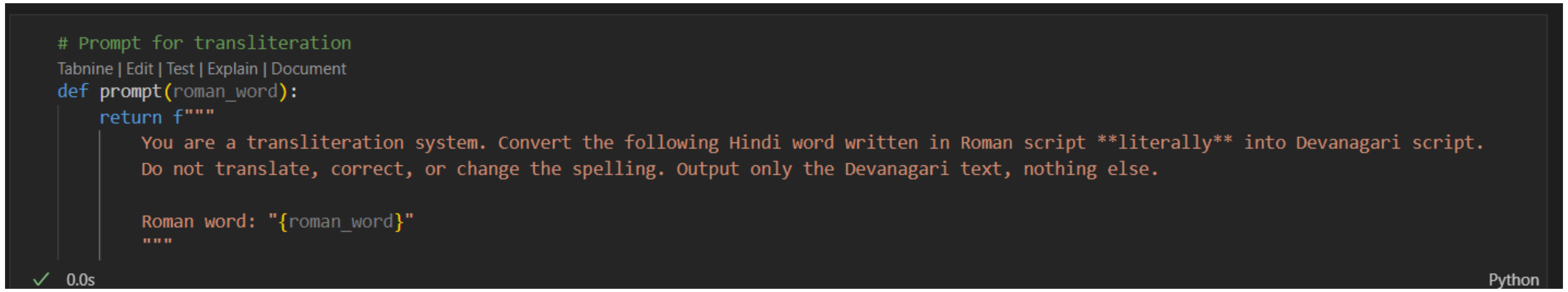
- Compare greedy decoding vs beam search
  Greedy:
  ```
  test Loss: 0.3020 | test ACC: 0.4967 | test F1: 0.8942
  ```
  Beam search:
  ```
  test Loss: 0.3020 | test ACC: 0.5211 | test F1: 0.9057
  ```
- Is your program running?
  yes

```
test Loss: 0.3220 | test ACC: 0.5011 | test F1: 0.8943
  test Loss: 0.3220 | test ACC: 0.5011 | test F1: 0.8943
    test Loss: 0.3220 | test ACC: 0.5011 | test F1: 0.8943
```

LLM Based Transliteration

What did you read for this part of the assignment?
see youtube videos how api calls is done

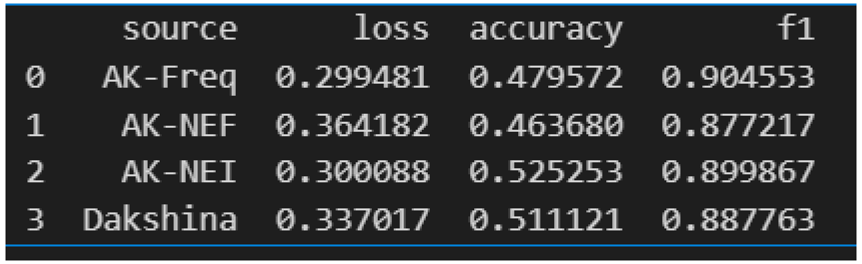
- Which LLM did you use?
  gpt-oss:120b-cloud
- Try with various values for temperature and top_p.
  no takes so much time with values
  Test ACC: 0.82, Test f1 score :0.28 (for only 2000 instances)
- Is your program running?
  yes



```python
# Prompt for transliteration
Tabnine | Edit | Test | Explain | Document
def prompt(roman_word):
    return f"""
        You are a transliteration system. Convert the following Hindi word written in Roman script **literally** into Devanagari script.
        Do not translate, correct, or change the spelling. Output only the Devanagari text, nothing else.

        Roman word: "{roman_word}"
        """
```
✓ 0.0s Python

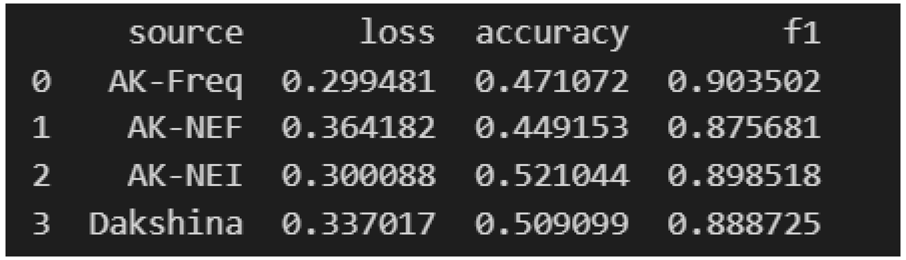
Error analysis based on source

Beam Search

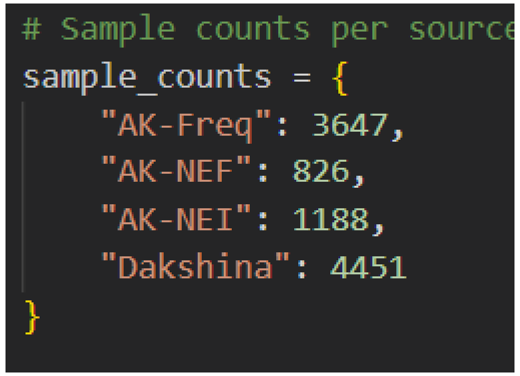


```
     source      loss  accuracy        f1
0   AK-Freq  0.299481  0.479572  0.904553
1    AK-NEF  0.364182  0.463680  0.877217
2    AK-NEI  0.300088  0.525253  0.899867
3  Dakshina  0.337017  0.511121  0.887763
```

Greedy

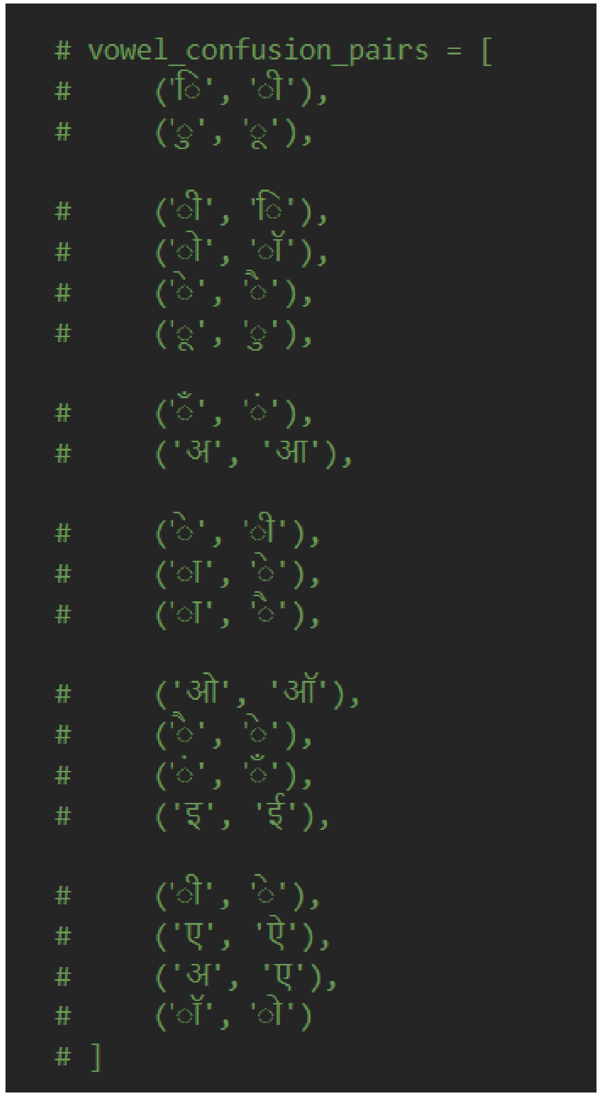


```
     source      loss  accuracy        f1
0   AK-Freq  0.299481  0.471072  0.903502
1    AK-NEF  0.364182  0.449153  0.875681
2    AK-NEI  0.300088  0.521044  0.898518
3  Dakshina  0.337017  0.509099  0.888725
```

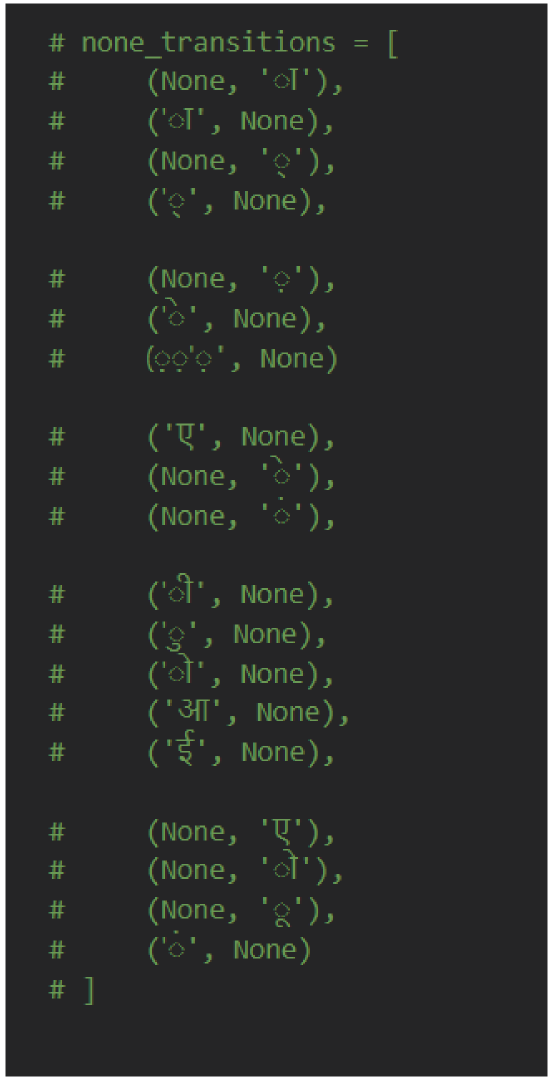


```
# Sample counts per source
sample_counts = {
    "AK-Freq": 3647,
    "AK-NEF": 826,
    "AK-NEI": 1188,
    "Dakshina": 4451
}
```

Character level Error analysis

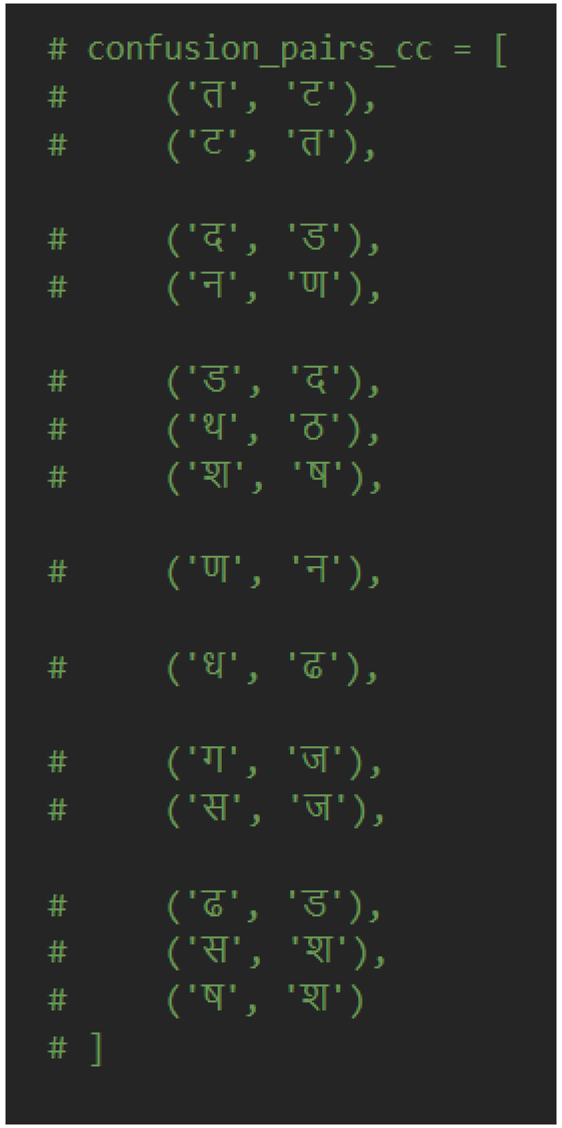


```
# vowel_confusion_pairs = [
#     ('ि', 'ी'),
#     ('ु', 'ू'),

#     ('ी', 'ि'),
#     ('ो', 'ॉ'),
#     ('े', 'ै'),
#     ('ू', 'ु'),

#     ('ँ', 'ं'),
#     ('अ', 'आ'),

#     ('े', 'ी'),
#     ('ा', 'े'),
#     ('ा', 'ै'),

#     ('ओ', 'ऑ'),
#     ('ै', 'े'),
#     ('ं', 'ँ'),
#     ('इ', 'ई'),

#     ('ी', 'े'),
#     ('ए', 'ऐ'),
#     ('अ', 'ए'),
#     ('ॉ', 'ो')
# ]
```

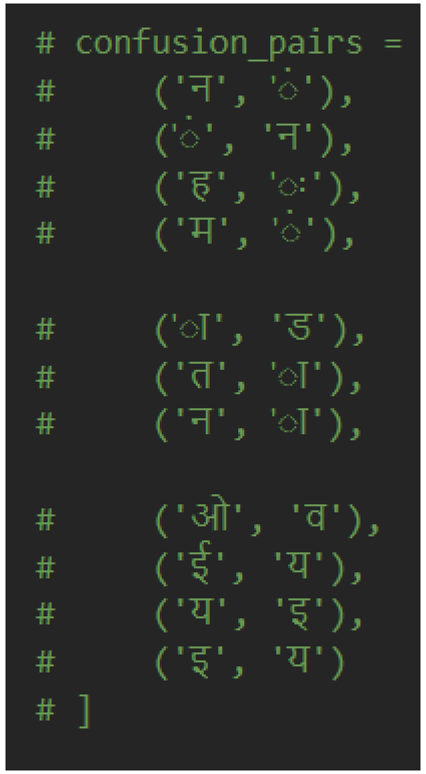


```
# none_transitions = [
#     (None, 'ा'),
#     ('ा', None),
#     (None, '्'),
#     ('्', None),

#     (None, '़'),
#     ('े', None),
#     (़़'़', None)

#     ('ए', None),
#     (None, 'े'),
#     (None, 'ं'),

#     ('ी', None),
#     ('ु', None),
#     ('ो', None),
#     ('आ', None),
#     ('ई', None),

#     (None, 'ए'),
#     (None, 'ो'),
#     (None, 'ू'),
#     ('ं', None)
# ]
```

Character level Error analysis



```
# confusion_pairs_cc = [
#     ('त', 'ट'),
#     ('ट', 'त'),

#     ('द', 'ड'),
#     ('न', 'ण'),

#     ('ड', 'द'),
#     ('थ', 'ठ'),
#     ('श', 'ष'),

#     ('ण', 'न'),

#     ('ध', 'ढ'),

#     ('ग', 'ज'),
#     ('स', 'ज'),

#     ('ढ', 'ड'),
#     ('स', 'श'),
#     ('ष', 'श')
# ]
```



```
# confusion_pairs =
#     ('न', 'ं'),
#     ('ं', 'न'),
#     ('ह', 'ः'),
#     ('म', 'ं'),

#     ('ा', 'ड'),
#     ('त', 'ा'),
#     ('न', 'ा'),

#     ('ओ', 'व'),
#     ('ई', 'य'),
#     ('य', 'इ'),
#     ('इ', 'य')
# ]
```