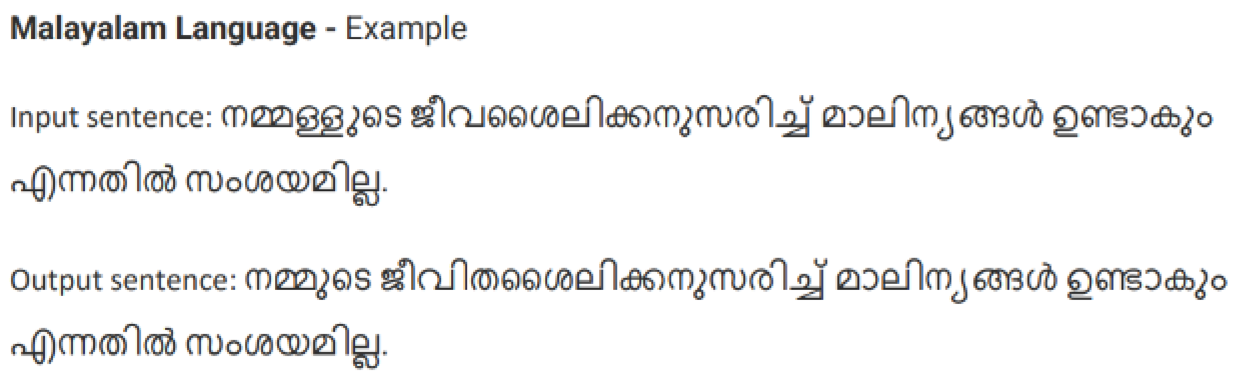
23BAI1097 ML Challenging Experiment

Task Description

Grammatical Error Correction (GEC) for one of the following low-resource languages: Bangla, Hindi, Malayalam, Tamil, Telugu

Malayalam has been chosen for this experiment

Metric of evaluation: GLEU Score (ranges between 0 to 1)



Malayalam Language - Example

Input sentence: നമ്മള്ളുടെ ജീവശൈലിക്കനുസരിച്ച് മാലിന്യങ്ങൾ ഉണ്ടാകും എന്നതിൽ സംശയമില്ല.

Output sentence: നമ്മുടെ ജീവിതശൈലിക്കനുസരിച്ച് മാലിന്യങ്ങൾ ഉണ്ടാകും എന്നതിൽ സംശയമില്ല.

image.png

Experiment Setup

```
!pip install sentencepiece
```

```
Requirement already satisfied: sentencepiece in /usr/local/lib/python3.12/dis
```

```
import numpy as np
import pandas as pd
import nltk
nltk.download('punkt')
from nltk.translate.gleu_score import sentence_gleu, corpus_gleu
import torch
from transformers import M2M100Config, M2M100ForConditionalGeneration, M2M100
from torch.optim import AdamW

if torch.cuda.is_available():
    device = torch.device("cuda")
else:
    device = torch.device("cpu")
```

```
# Upload datasets
from google.colab import files
files.upload()
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

```
 <input type="file" id="files-2bfea0df-751d-49c5-8aca-ccd677919358" name="fil
    style="border:none" />
 <output id="result-2bfea0df-751d-49c5-8aca-ccd677919358">
  Upload widget is only available when the cell has been executed in the
  current browser session. Please rerun this cell to enable.
  </output>
  <script>// Copyright 2017 Google LLC
```

/** * @fileoverview Helpers for google.colab Python module. */ (function(scope) { function span(text, styleAttributes = {}) { const element = document.createElement(‘span’); element.textContent = text; for (const key of Object.keys(styleAttributes)) { element.style[key] = styleAttributes[key]; } return element; }

// Max number of bytes which will be uploaded at a time. const MAX_PAYLOAD_SIZE = 100 * 1024;

function _uploadFiles(inputId, outputId) { const steps = uploadFilesStep(inputId, outputId); const outputElement = document.getElementById(outputId); // Cache steps on the outputElement to make it available for the next call // to uploadFilesContinue from Python. outputElement.steps = steps;

return _uploadFilesContinue(outputId); }

// This is roughly an async generator (not supported in the browser yet), // where there are multiple asynchronous steps and the Python side is going // to poll for completion of each step. // This uses a Promise to block the python side on completion of each step, // then passes the result of the previous step as the input to the next step. function _uploadFilesContinue(outputId) { const outputElement = document.getElementById(outputId); const steps = outputElement.steps;

const next = steps.next(outputElement.lastPromiseValue); return Promise.resolve(next.value.promise).then((value) => { // Cache the last promise value to make it available to the next // step of the generator. outputElement.lastPromiseValue = value; return next.value.response; }); }

/** * Generator function which is called between each async step of the upload * process. * @param {string} inputId Element ID of the input file picker element. * @param {string} outputId Element ID of the output display. * @return {!Iterable<!Object>} Iterable of next steps. / *function* uploadFilesStep(inputId, outputId) { const inputElement = document.getElementById(inputId); inputElement.disabled = false;

const outputElement = document.getElementById(outputId); outputElement.innerHTML = '';

const pickedPromise = new Promise((resolve) => { inputElement.addEventListener('change', (e) => { resolve(e.target.files); }); });

const cancel = document.createElement('button'); inputElement.parentElement.appendChild(cancel); cancel.textContent = 'Cancel upload'; const cancelPromise = new Promise((resolve) => { cancel.onclick = () => { resolve(null); }; });

// Wait for the user to pick the files. const files = yield { promise: Promise.race([pickedPromise, cancelPromise]), response: { action: 'starting', } };

cancel.remove();

// Disable the input element since further picks are not allowed. inputElement.disabled = true;

if (!files) { return { response: { action: 'complete', } }; }

for (const file of files) { const li = document.createElement('li'); li.append(span(file.name, {fontWeight: 'bold'})); li.append(span(`(${file.type || 'n/a'}) - ${file.size} bytes, + last modified: ${ file.lastModifiedDate ? file.lastModifiedDate.toLocaleDateString() : 'n/a'} -`)); const percent = span('0% done'); li.appendChild(percent);

```
outputElement.appendChild(li);

const fileDataPromise = new Promise((resolve) => {
  const reader = new FileReader();
  reader.onload = (e) => {
    resolve(e.target.result);
  };
  reader.readAsArrayBuffer(file);
});
// Wait for the data to be ready.
```

```
let fileData = yield {
  promise: fileDataPromise,
  response: {
    action: 'continue',
  }
};

// Use a chunked sending to avoid message size limits. See b/62115660.
let position = 0;
do {
  const length = Math.min(fileData.byteLength - position, MAX_PAYLOAD_SIZE);
  const chunk = new Uint8Array(fileData, position, length);
  position += length;

  const base64 = btoa(String.fromCharCode.apply(null, chunk));
  yield {
    response: {
      action: 'append',
      file: file.name,
      data: base64,
    },
  };

  let percentDone = fileData.byteLength === 0 ?
      100 :
      Math.round((position / fileData.byteLength) * 100);
  percent.textContent = `${percentDone}% done`;

} while (position < fileData.byteLength);

}
```

// All done. yield { response: { action: ‘complete’, } }; }

scope.google = scope.google || {}; scope.google.colab = scope.google.colab || {};
scope.google.colab._files = { _uploadFiles, _uploadFilesContinue, }; })(self);

```
Saving dev.csv to dev.csv
Saving train.csv to train.csv
```

```
{'dev.csv': b'Input sentence,Output sentence\n\xe0\xae\xae\xe0\xaf\x81\xe0\xae
 'train.csv': b'Input sentence,Output sentence\r\n"\xe0\xae\x8e\xe0\xae\xa9\xe
```

```
train_set = pd.read_csv("/content/train.csv")
test_set = pd.read_csv("/content/dev.csv")
```

```
print(train_set.columns)
print(len(train_set))

Index(['Input sentence', 'Output sentence'], dtype='object')
91

train_set.dropna(inplace=True)
test_set.dropna(inplace=True)

X = train_set.iloc[:, 0].values
y = train_set.iloc[:, 1].values

X_val = test_set.iloc[:, 0].values
y_val = test_set.iloc[:, 1].values

model = M2M100ForConditionalGeneration.from_pretrained("facebook/m2m100_418M"
tokenizer = M2M100Tokenizer.from_pretrained("facebook/m2m100_418M", src_lang

config.json:   0%|          | 0.00/908 [00:00<?, ?B/s]



pytorch_model.bin:   0%|          | 0.00/1.94G [00:00<?, ?B/s]



model.safetensors:   0%|          | 0.00/1.94G [00:00<?, ?B/s]



generation_config.json:   0%|          | 0.00/233 [00:00<?, ?B/s]



tokenizer_config.json:   0%|          | 0.00/298 [00:00<?, ?B/s]



vocab.json: 0.00B [00:00, ?B/s]



sentencepiece.bpe.model:   0%|          | 0.00/2.42M [00:00<?, ?B/s]



special_tokens_map.json: 0.00B [00:00, ?B/s]

model.to(device)
model.train()

optimizer = AdamW(model.parameters(), lr=2e-5)
batch_size = 4

for i in range(0, len(X), batch_size):
  inputs = X[i: i+batch_size]
  targets = y[i: i+batch_size]

  # return_tensors = "pt" ensures the output is a pytorch tensor
  model_inputs = tokenizer(list(inputs), text_target=list(targets), padding=T

  outputs = model(**model_inputs)
  loss = outputs.loss
  loss.backward()
  optimizer.step()
  optimizer.zero_grad()

preds = []
refs = []

model.eval()
for inp, ref in list(zip(X_val, y_val)):
    encoded = tokenizer(inp, return_tensors="pt").to(model.device)
    generated = model.generate(**encoded, forced_bos_token_id=tokenizer.get_la
    pred = tokenizer.decode(generated[0], skip_special_tokens=True)
    preds.append([pred.split()])
    refs.append(ref.split())


print(preds)
print(refs)

[[['��������', '����������']], [['-�������/������', '�������������
[['��������', '����������'], ['-�������/�����', '���������������'

score = corpus_gleu(preds, refs)
print("GLEU Score:", score)

GLEU Score: 0.46938775510204084
```