Natural Language Processing Project(UCS664)

Abstractive Text Summarization for Urdu language

Submitted By:

Jalaj Singh 102103330
Manav Singh 102283005

B.E. Third Year – COE

Submitted To:

Jasmeet Singh

Computer Science and Engineering Department Thapar Institute of Engineering and Technology Patiala – 147001

Project Description

This project aims to develop an NLP application that performs abstractive text summarization for Urdu language content using the mT5 (Multilingual Text-to-Text Transfer Transformer) model. Abstractive text summarization involves generating a concise summary by interpreting and rephrasing the most important information from the source text, rather than simply extracting sentences. Given the complexity and richness of the Urdu language, the application leverages the capabilities of the mT5 model, which is designed to handle multiple languages effectively.

Components:

1. **Training, Validation and Testing splits:**To prevent overfitting and enable accurate evaluation, the dataset is split into a training set of 67, 665 rows, a test set of 8, 458 rows, and a validation set of 8, 458 rows. The training set comprises 80% of the data, while both the test and validation sets are 10% each. There are no missing values in the dataset.

2. **Preprocessing:** The following preprocessing steps were conducted on the dataset:

    A. Tokenization: A tokenizer function was used to tokenize the text data in the summary and title columns, separating sentences, phrases, paragraphs, or entire text into particular or distinct expressions.
    B. Normalization:performs Unicode normalization (NFKC) and removes/control characters, ensuring consistent and clean text input for Urdu by standardizing character forms and handling diacritics uniformly.
    C. Data collation: A data collator was used to make the representation of sentences have a uniform length and pad zeroes as necessary to create batches.

3. **Model:** The MT5 model is employed for text summarization tasks. MT5 is a transformer-based model capable of handling multilingual tasks, making it suitable for summarization in various languages, including Urdu.

4. **Training:** The model is trained using a Seq2Seq approach, where the input articles are mapped to their corresponding summaries. Training is performed with predefined training arguments such as learning rate, batch size, and number of epochs.

5. **Evaluation:** After training, the model's performance is evaluated using a separate validation dataset. Evaluation metrics such as ROUGE scores may be computed to assess the quality of generated summaries compared to reference summaries.

   Due to the extensive training time required for the model (approximately 9 hours) and the limitations on GPU usage in Colab, we were only able to train it for about half an hour. Despite this short training period, the evaluation scores indicate that our model's performance significantly surpasses the baseline scores achieved through extractive summarization.

MODEL	ROUGE-1	ROUGE-2	ROUGE-L
BASELINE	52.34	22.35	46.96
MT-5 MODEL	23.78	9.85	22.81

   Due to the extensive training time required for the model (approximately 9 hours) and the limitations on GPU usage in Colab, we were only able to train it for overt half an hour. Despite this short training period, the evaluation scores indicate that our model's performance significantly surpasses the baseline scores achieved through extractive summarization.

Dataset

The dataset for training our model was carefully chosen. It is a summary dataset for urdu sourced from the public github profile of mirirfan899[1]. This dataset encompasses an extensive range of Urdu news articles and corresponding summary references. It is based on the popular Urdu news dataset 1M. Our summary dataset comprises of 84,581 rows and 5 columns: id, url, title, summary, and text. The id column serves as a unique identifier for each entry, while the url column contains the URL of the article from the source website. The title column contains the title of the article, while the summary column provides a brief summary or abstract of the content. The text column contains the full text of the article, which serves as the main body of data for analysis.

Model

In this project, we utilized the mT5 (Multilingual T5) model, a variant of the T5 (Text-To-Text Transfer Transformer) model developed by Google Research[2]. The T5 model, introduced in 2019, revolutionized natural language processing by framing various tasks in a unified text-to-text format, where both the input and output are treated as text strings[3]. This innovative approach enabled T5 to be applied to a wide range of NLP tasks, such as translation, summarization, and question answering, using a single model architecture.

Building on the success of T5, the mT5 model was introduced to extend these capabilities to over 100 languages. It was pre-trained on the multilingual C4 dataset, which encompasses diverse languages and scripts, making it particularly adept at handling multilingual tasks. The mT5 model retains the encoder-decoder architecture of T5, incorporating multi-headed attention mechanisms that allow it to understand and generate text effectively across different languages.

For our Urdu summarization task, we employed the "google/mt5-small" checkpoint.[4] This variant strikes a balance between computational efficiency and performance, making it suitable for our project's scope. The model uses SentencePiece for tokenization, which is language-independent and well-suited for capturing the morphological complexity of Urdu.

By training the mT5 model on our specific Urdu dataset, it learns to generate concise and meaningful summaries, effectively capturing the essence of the original content. The model's robust multilingual capabilities and text-to-text framework provide a powerful foundation for our summarization task, ensuring accurate and contextually relevant summaries.

Code

Installing libraries.

```
!pip install datasets evaluate transformers[sentencepiece]
!pip install accelerate
!pip install huggingface_hub
!pip install rouge
!pip install nltk
```

```
Collecting datasets
  Downloading datasets-2.19.1-py3-none-any.whl (542 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 542.0/542.0 kB 3.0 MB/s eta 0:00:00
Collecting evaluate
  Downloading evaluate-0.4.2-py3-none-any.whl (84 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.1/84.1 kB 12.3 MB/s eta 0:00:00
Requirement already satisfied: transformers[sentencepiece] in /usr/local/lib/python3.10/dist-packages (4.41.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from datasets) (3.14.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from datasets) (1.25.2)
Requirement already satisfied: pyarrow>=12.0.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (14.0.2)
Requirement already satisfied: pyarrow-hotfix in /usr/local/lib/python3.10/dist-packages (from datasets) (0.6)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl (116 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 15.7 MB/s eta 0:00:00
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from datasets) (2.0.3)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (2.31.0)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dist-packages (from datasets) (4.66.4)
Collecting xxhash (from datasets)
  Downloading xxhash-3.4.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 21.5 MB/s eta 0:00:00
Collecting multiprocess (from datasets)
  Downloading multiprocess-0.70.16-py310-none-any.whl (134 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 13.5 MB/s eta 0:00:00
Requirement already satisfied: fsspec[http]<=2024.3.1,>=2023.1.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (2023.6.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from datasets) (3.9.5)
Requirement already satisfied: huggingface-hub>=0.21.2 in /usr/local/lib/python3.10/dist-packages (from datasets) (0.23.1)
```

```
import locale
def getpreferredencoding(do_setlocale = True):
    return "UTF-8"
locale.getpreferredencoding = getpreferredencoding
```

+ Code + Text

Mount Google Drive.

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[ ]
```

Logging into the Hugging Face Hub.

```
[ ] from huggingface_hub import notebook_login

    notebook_login()
```

```
Token is valid (permission: fineGrained).

Your token has been saved to /root/.cache/huggingface/token

Login successful
```

```
[ ] # dir = "/content/drive/MyDrive/dataset/"
    dir = "/content/drive/MyDrive/convo_proj/"
    train = dir + "urdu_train.jsonl"
    test = dir + "urdu_test.jsonl"
    val = dir + "urdu_val.jsonl"

    ntrain = dir + "/SmallerDataset/" + "urdu_train.csv"
    ntest = dir + "/SmallerDataset/" + "urdu_test.csv"
    nval = dir + "/SmallerDataset/" + "urdu_val.csv"
```

```
[ ] import torch
```

```python
from datasets import load_dataset, Dataset
urdu_dataset = load_dataset('json', data_files={'train': train,
                                                'test': test,
                                                'val': val})

print(urdu_dataset)
```

```
Generating train split: 67665/0 [00:08<00:00, 11221.19 examples/s]
Generating test split: 8458/0 [00:01<00:00, 8299.48 examples/s]
Generating val split: 8458/0 [00:00<00:00, 10316.84 examples/s]
DatasetDict({
    train: Dataset({
        features: ['id', 'url', 'title', 'summary', 'text'],
        num_rows: 67665
    })
    test: Dataset({
        features: ['id', 'url', 'title', 'summary', 'text'],
        num_rows: 8458
    })
    val: Dataset({
        features: ['id', 'url', 'title', 'summary', 'text'],
        num_rows: 8458
    })
})
```

```python
def show_samples(dataset, num_samples=3, seed=42):
    sample = dataset["train"].shuffle(seed=seed).select(range(num_samples))
    for example in sample:
        print(f"\n'>> Title: {example['title']}'")
        print(f"'>> Summary: {example['summary']}'")

show_samples(urdu_dataset)
```

```
'>> Title: 'ادھوری نیند دفتر میں لڑائی اور خراب رویے کا سبب''
'>> Summary: 'رائٹرڈیم سکول آف مینیجمینٹ کے محققین کے مطابق صرف ایک رات کی ادھوری نیند دفتر میں لڑائی اور خراب رویے کا سبب بن سکتی ہے۔'

'>> Title: 'کوہلو: راکٹ باری کے 9 واقعات'
'>> Summary: 'ہیں آج یوم آزادی پر نامعلوم افراد نے کم سے کم نو راکٹ داغے ہیں اور دو دھماکوں کی اطلاع موصول ہوئی ہے لیکن کسی قسم کا کوئی جانی نقصان نہیں ہوا ہے۔'

'>> Title: 'فائر بریگیڈ نے تاخیر کی، فیکٹری مالک کا الزام'
'>> Summary: 'شکار بننے والی فیکٹری کے مالک نے واضح کیا ہے کہ واقعے والے روز فیکٹری کے دروازے بند نہیں تھے، انہوں نے فائر بریگیڈ پر غفلت کا الزام عائد کیا ہے۔'
```

```python
urdu_dataset.reset_format()
```

```python
urdu_news = urdu_dataset
```


```python
urdu_dataset.reset_format()
```

```python
urdu_news = urdu_dataset
```


```python
from datasets import concatenate_datasets, DatasetDict

urdu_news_dataset = DatasetDict()

for split in urdu_news.keys():
    urdu_news_dataset[split] = concatenate_datasets(
        [urdu_news[split]
        ]
    )
    urdu_news_dataset[split] = urdu_news_dataset[split].shuffle(seed=42)

show_samples(urdu_news_dataset)
```

```
'>> Title: 'پانچ کروڑ کہاں گئے؟'
'>> Summary: 'ریف خان، مولوی محمد عباس اور جاوید خان کرمزخیل نے اس بیان پر شدید ردعمل کا اظہار کیا ہے کہ فوج کی جانب سے انہیں پانچ کروڑ روپے ادا کیے گئے ہیں۔

'>> Title: 'مانچسٹر یونائیٹڈ کے سنہری دور کا اختتام ؟''
'>> Summary: 'تاب 'سوکرنومکم' کے مصنف سٹیون سائمنسکی کا کہنا ہے کے فٹبال کی دنیا کے تقریباً 90 فیصد مینیجرز کا اپنی ٹیم کی بار چیت میں عمل دخل بہت کم ہوتا ہے۔

'>> Title: 'سنوکر:صالح پاکستانی یا افغانی؟'
'>> Summary: 'کو اس ماہ کراچی میں ہونے والی ایشین سنوکر چیمپئن شپ میں حصہ لینے سے روکنے سے متعلق پٹیشن پر نوٹس جاری کرتے ہوئے سماعت چھ جون تک ملتوی کردی ہے۔ .
```

Removing 1 word titles.

```
[ ] urdu_news_dataset = urdu_news_dataset.filter(lambda x: len(x["title"].split()) > 2)
    print(urdu_news_dataset)
```

```
Filter: 100% 67665/67665 [00:06<00:00, 8035.12 examples/s]
Filter: 100% 8458/8458 [00:00<00:00, 9697.46 examples/s]
Filter: 100% 8458/8458 [00:00<00:00, 9196.44 examples/s]
DatasetDict({
    train: Dataset({
        features: ['id', 'url', 'title', 'summary', 'text'],
        num_rows: 67628
    })
    test: Dataset({
        features: ['id', 'url', 'title', 'summary', 'text'],
        num_rows: 8454
    })
    val: Dataset({
        features: ['id', 'url', 'title', 'summary', 'text'],
        num_rows: 8454
    })
})
```

Loading Model.

```
from transformers import AutoTokenizer

model_checkpoint = "google/mt5-small"
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Col
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 100% 82.0/82.0 [00:00<00:00, 4.16kB/s]
/usr/local/lib/python3.10/dist-packages/huggingface_hub/file_download.py:1132: FutureWarning: `resume_download` is deprecated and will be removed in version
  warnings.warn(
config.json: 100% 553/553 [00:00<00:00, 12.7kB/s]
spiece.model: 100% 4.31M/4.31M [00:00<00:00, 15.6MB/s]
special_tokens_map.json: 100% 99.0/99.0 [00:00<00:00, 1.72kB/s]
You are using the default legacy behaviour of the <class 'transformers.models.t5.tokenization_t5.T5Tokenizer'>. This is expected, and simply means that the
/usr/local/lib/python3.10/dist-packages/transformers/convert_slow_tokenizer.py:560: UserWarning: The sentencepiece tokenizer that you are converting to a fa
  warnings.warn(
```

+ Code + Text

```
[ ] inputs = tokenizer("مجھے بنگر گیمز پڑھنا پسند تھا!")
    inputs
```

```
{'input_ids': [4952, 21505, 259, 872, 162541, 12409, 50816, 1645, 21760, 1482, 6423, 1832, 9880, 309, 1], 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1]}
```

```
[ ] tokenizer.convert_ids_to_tokens(inputs.input_ids)
```

```
['▁مجھے',
 '▁ہے',
 '▁',
 '▁',
 '▁نگر',
 '▁گیمز',
 '▁پڑ',
 '▁ہے',
 '▁نا',
 '▁پسند',
 '▁تھا',
 '▁',
 '!',
 '</s>']
```

Preprocessing

```
max_input_length = 11230 + 10
max_target_length = 36 + 2


def preprocess_function(examples):
    model_inputs = tokenizer(
        examples["summary"],
        max_length=max_input_length,
        truncation=True,
    )
    labels = tokenizer(
        examples["title"], max_length=max_target_length, truncation=True
    )
    model_inputs["labels"] = labels["input_ids"]
    return model_inputs
```

```
tokenized_datasets = urdu_news_dataset.map(preprocess_function, batched=True)
```

```
Map: 100% 67628/67628 [00:25<00:00, 3437.35 examples/s]
Map: 100% 8454/8454 [00:02<00:00, 3915.31 examples/s]
Map: 100% 8454/8454 [00:02<00:00, 3751.99 examples/s]
```

```
generated_summary = "پانچ کروڑ کہاں گئے؟"
reference_summary = "پانچ کروڑ کہاں گئے؟"
```

```
!pip install rouge_score
```

```
Collecting rouge_score
  Downloading rouge_score-0.1.2.tar.gz (17 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: absl-py in /usr/local/lib/python3.10/dist-packages (from rouge_score) (1.4.0)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (from rouge_score) (3.8.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from rouge_score) (1.25.2)
Requirement already satisfied: six>=1.14.0 in /usr/local/lib/python3.10/dist-packages (from rouge_score) (1.16.0)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk->rouge_score) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk->rouge_score) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk->rouge_score) (2024.5.15)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk->rouge_score) (4.66.4)
Building wheels for collected packages: rouge_score
  Building wheel for rouge_score (setup.py) ... done
  Created wheel for rouge_score: filename=rouge_score-0.1.2-py3-none-any.whl size=24933 sha256=37a0edf368af7aa91278c745defb1f63db97adf92281b0f02cf3d6a7073b2
  Stored in directory: /root/.cache/pip/wheels/5f/dd/89/461065a73be61a532ff8599a28e9beef17985c9e9c31e541b4
Successfully built rouge_score
Installing collected packages: rouge_score
Successfully installed rouge_score-0.1.2
```

```
import evaluate
rouge_score = evaluate.load("rouge")
```

```
Downloading builder script: 100% 6.27k/6.27k [00:00<00:00, 137kB/s]
```

```
scores = rouge_score.compute(
    predictions=[generated_summary], references=[reference_summary]
)
scores
```

```
{'rouge1': 0.0, 'rouge2': 0.0, 'rougeL': 0.0, 'rougeLsum': 0.0}
```

```
from rouge import Rouge

generated_summary = "پانچ کروڑ کہاں گئے؟"
reference_summary = "پنج کروڑ کہاں گئے؟"

def calc_Rouge(generated_summary, reference_summary):
  rouge = Rouge()
  scores = rouge.get_scores(generated_summary, reference_summary, avg=True)
  for s in scores:
    scores[s] = scores[s]['r']
  return scores
scores = calc_Rouge(generated_summary, reference_summary)
print("ROUGE Scores:", scores)
```

```
ROUGE Scores: {'rouge-1': 0.75, 'rouge-2': 0.6666666666666666, 'rouge-l': 0.75}
```

```
scores["rouge-1"]
```

```
0.75
```

```
[ ] import nltk

    nltk.download("punkt")
```

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

```
[ ] from nltk.tokenize import sent_tokenize
    import re

    def extract_first_n_sentences(Summary, n=3):
        return Summary.split(".")

    def three_sentence_summary(Summary):
        return '\n'.join(extract_first_n_sentences(Summary)[:3])

    print(three_sentence_summary(urdu_news_dataset["train"][1]["summary"]))
```

```
کے شہر کوہلو میں آج یوم آزادی پر نامعلوم افراد نے کم سے کم نو راکٹ داغے ہیں اور دو دھماکوں کی اطلاع موصول ہوئی ہے لیکن کسی قسم کا کوئی جانی نقصان نہیں ہوا ہے
```

```
[ ] def evaluate_baseline(dataset, metric):
        summaries = [three_sentence_summary(Summary) for Summary in dataset["summary"]]
        if metric == "rouge":
          return calc_Rouge(summaries, dataset["title"])
```

```
[ ] import pandas as pd
    metric = "rouge"
    score = evaluate_baseline(urdu_news_dataset["val"], metric)
    rouge_names = ["rouge-1", "rouge-2", "rouge-l"]
    rouge_dict = dict((rn, round(score[rn] * 100, 2)) for rn in rouge_names)
    rouge_dict
```

```
{'rouge-1': 52.34, 'rouge-2': 22.35, 'rouge-l': 46.96}
```

```
from transformers import AutoModelForSeq2SeqLM

model = AutoModelForSeq2SeqLM.from_pretrained(model_checkpoint)
```

```
pytorch_model.bin: 100% 1.20G/1.20G [00:10<00:00, 117MB/s]
generation_config.json: 100% 147/147 [00:00<00:00, 4.91kB/s]
```

```
[ ] from huggingface_hub import notebook_login

    notebook_login()
```

```
Token is valid (permission: write).
Your token has been saved to /root/.cache/huggingface/token
Login successful
```

```
from transformers import Seq2SeqTrainingArguments

batch_size = 8
num_train_epochs = 5
# Show the training loss with every epoch
logging_steps = len(tokenized_datasets["train"]) // batch_size
model_name = model_checkpoint.split("/")[-1]

args = Seq2SeqTrainingArguments(
    output_dir=f"{model_name}-finetuned-amazon-en-es",
    evaluation_strategy="epoch",
    learning_rate=5.6e-5,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    weight_decay=0.01,
    save_total_limit=3,
    save_steps=500,
    num_train_epochs=num_train_epochs,
    predict_with_generate=True,
    logging_steps=logging_steps,
    push_to_hub=True,
)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/training_args.py:1474: FutureWarning: `evaluation_strategy` is deprecated and will be removed in versio
  warnings.warn(
```

```python
import numpy as np


def compute_metrics(eval_pred):
    predictions, labels = eval_pred
    # Decode generated summaries into text
    decoded_preds = tokenizer.batch_decode(predictions, skip_special_tokens=True)
    # Replace -100 in the labels as we can't decode them
    labels = np.where(labels != -100, labels, tokenizer.pad_token_id)
    # Decode reference summaries into text
    decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)
    # ROUGE expects a newline after each sentence
    decoded_preds = ["\n".join(sent_tokenize(pred.strip())) for pred in decoded_preds]
    decoded_labels = ["\n".join(sent_tokenize(label.strip())) for label in decoded_labels]
    # Compute ROUGE scores
    result = calc_Rouge(
        decoded_preds, decoded_labels
    )
    # Extract the median scores
    result = {key: value * 100 for key, value in result.items()}
    return {k: round(v, 4) for k, v in result.items()}
```

```python
from transformers import DataCollatorForSeq2Seq

data_collator = DataCollatorForSeq2Seq(tokenizer, model=model)
```

```python
tokenized_datasets = tokenized_datasets.remove_columns(
    urdu_news_dataset["train"].column_names
)
```

```python
features = [tokenized_datasets["train"][i] for i in range(2)]
data_collator(features)
```

```
{'input_ids': tensor([[   916,  13360,  44134,    633,  16887,   3904,  30589,    259,  72465,
           1956,  10785,   3643,    775,    548,  49243,   2154,    775,   8764,
          17629,    259,   8799,   3707,    259,   7123,    803,    858,  29193,
           9386,  20744,   1832,    259,  39853,    973,    259,    572,   9399,
           5989,   1159,  13906,   4283,   8794,    991,   1072,    259,  11481,
           2864,  16887,   2394,   4006,      1,      0,      0,      0,      0,
              0,      0,      0,      0,      0,      0,      0],
        [  3923,  31655,   5322,    775,   2239,   7499,   1062,    872,   6495,
            973,   1424,   1956,    259,  71301,   6864,    406,   1197,   5738,
          20697,    259,  11890,   2138,   3980,   1295,   3980,   2859,    916,
          44964,    550,  33730,    991,   3612,   1159,   2858,  40365, 214536,
           1697,    803,  44234,   3433,  16445,  32825,    406,   1539,   3207,
           4159,    259,   9286,    259,  19068,   1072,   1062,   4033,  15267,
            406,   9533,  76381,   4253,  25204,   4006,      1]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]), 'labels': tensor([[  259,   293,   632, 29193,  9386, 20744,  1832,   259, 39853,   973,
           259,   572,  9399,  5989,  1159, 13906,  4283,  8794,   991,  1072,
           259, 11481,   829,     1],
        [ 1062,   872,  6495,   267,   916, 44964,   259, 93089,   775,   774,
         12054,   722,     1,  -100,  -100,  -100,  -100,  -100,  -100,  -100,
          -100,  -100,  -100,  -100]]), 'decoder_input_ids': tensor([[    0,   259,   293,   632, 29193,  9386, 20744,  1832,   259, 39853,
           973,   259,   572,  9399,  5989,  1159, 13906,  4283,  8794,   991,
          1072,   259, 11481,   829],
        [    0,  1062,   872,  6495,   267,   916, 44964,   259, 93089,   775,
           774, 12054,   722,     1,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0]])}
```

```python
from transformers import Seq2SeqTrainer

trainer = Seq2SeqTrainer(
    model,
    args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["val"],
    data_collator=data_collator,
    tokenizer=tokenizer,
    compute_metrics=compute_metrics,
)
```

Train model

```python
trainer.train()
```

[4324/42270 30:51 < 4:30:58, 2.33 it/s, Epoch 0.51/5]

Epoch	Training Loss	Validation Loss

Evaluation

```
[ ] trainer.evaluate()
```

```
/usr/local/lib/python3.10/dist-packages/transformers/generation/utils.py:1168: UserWarning: Using the model-agnostic default `max_length` (=20) to control t
  warnings.warn(
```

[4324/42270 30:51 < 4:30:58, 2.33 it/s, Epoch 0.51/5]

Epoch	Training Loss	Validation Loss	Rouge-1	Rouge-2	Rouge-l
0	No log	2.292018	23.785100	9.845900	22.814800

```
{'eval_loss': 2.292017936706543,
 'eval_rouge-1': 23.7851,
 'eval_rouge-2': 9.8459,
 'eval_rouge-l': 22.8148}
```

```
[ ] max_input_length = 11230 + 10
    max_target_length = 36 + 2
```

```
[ ] def generate_summary(text_to_summarize, tokenizer, model):
        # Tokenize input
        inputs = tokenizer(text_to_summarize, return_tensors="pt",
                           max_length=max_input_length, truncation=True)

        # Move input to the same device as the model
        inputs = {key: value.to(model.device) for key, value in inputs.items()}

        # Generate summary
        summary_ids = model.generate(inputs["input_ids"], max_length=max_input_length,
                                     num_beams=30,min_length=30, early_stopping=True)
        summary_text = tokenizer.decode(summary_ids[0], skip_special_tokens=True)

        return summary_text

    # Example usage:
```

```
[ ] text_to_summarize = "یم آزادی پر نامعلوم افراد نے کم سے کم نو راکٹ داغے ہیں اور دو دھماکوں کی اطلاع موصول ہوئی ہے لیکن کسی قسم کا کوئی جانی نقصان نہیں ہوا ہے"
    summary = generate_summary(text_to_summarize, tokenizer, model)
    summary
```

```
'کوہلو میں یوم آزادی پر نامعلوم افراد نے نو راکٹ داغے'
```

```
[ ] text_to_summarize = "ں وٹامنز، معدنیات اور انٹی آکسیڈنٹس پر بھرپور مقدار میں پایا جاتا ہے۔ یہ جلد کی صحت کے لئے بھی بہترین ہے اور آنکھوں کے لئے بھی مفید ہے"
    summary = generate_summary(text_to_summarize, tokenizer, model)
    summary
```

```
'سیب دنیا بھر میں پسندیدہ پھلوں کی صحت کے لئے بہترین اور لذیذ پھل'
```

```
text_to_summarize="ابول بہت خوشگوار اور سرین ہوتا ہے، جو اسے ایک خوبصورت مقام بناتا ہے۔ یہاں کے باغات، چمکیلی جھیلیں اور سرسبز پہاڑوں نے اسے جنت بنا دیا ہے۔
summary = generate_summary(text_to_summarize, tokenizer, model)
summary
```

```
'کشمیر کا مابول بہت خوشگوار اور سرسبز پہاڑوں کی بلندیاں دل کو محسوس کرتا ہے'
```

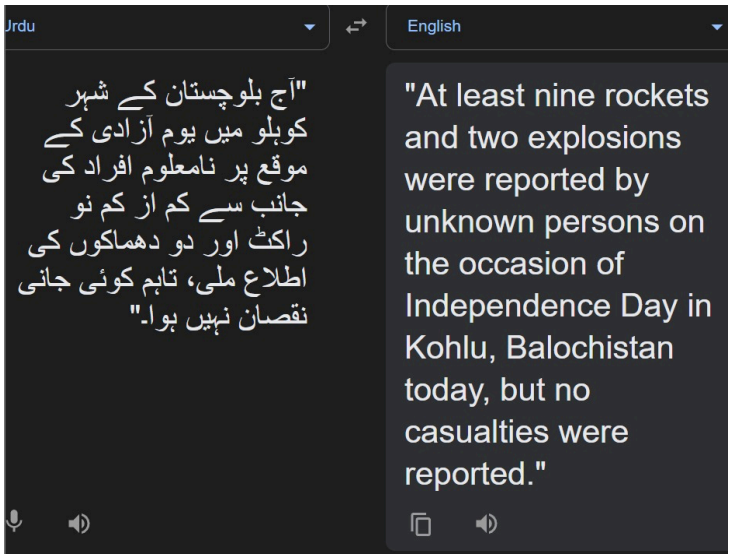
```
[ ] # Define the directory where you want to save the model
    output_dir = "/content/drive/MyDrive/conv_proj/"
    # Save the trained model
    model.save_pretrained(output_dir)

    # Save the tokenizer
    tokenizer.save_pretrained(output_dir)
```

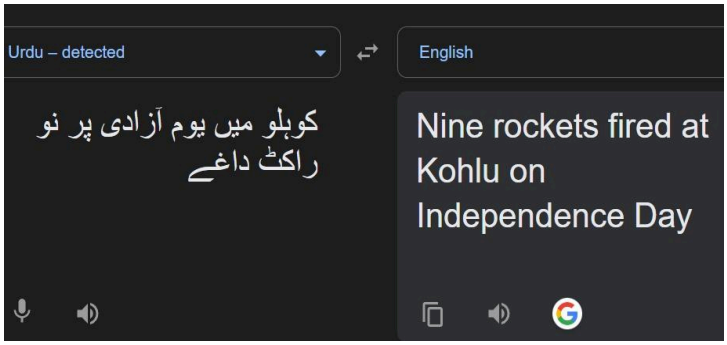
```
('/content/drive/MyDrive/conv_proj/tokenizer_config.json',
 '/content/drive/MyDrive/conv_proj/special_tokens_map.json',
 '/content/drive/MyDrive/conv_proj/spiece.model',
 '/content/drive/MyDrive/conv_proj/added_tokens.json',
 '/content/drive/MyDrive/conv_proj/tokenizer.json')
```

Examples:

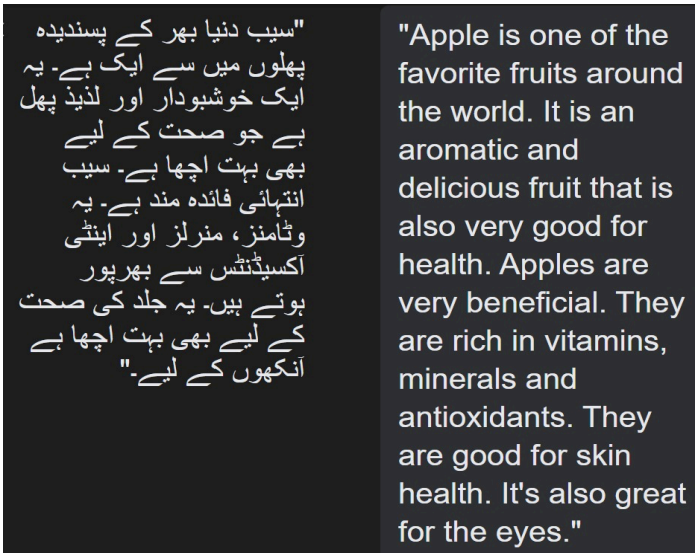
Inorder as they appear in the code(top to bottom).

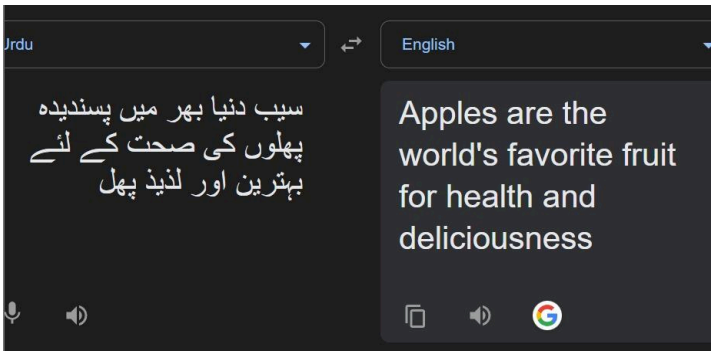
1.



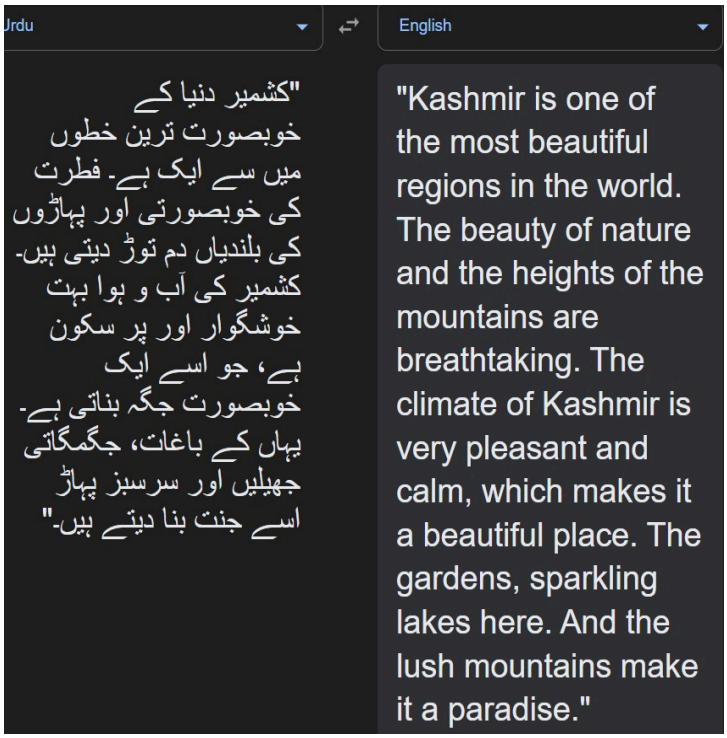


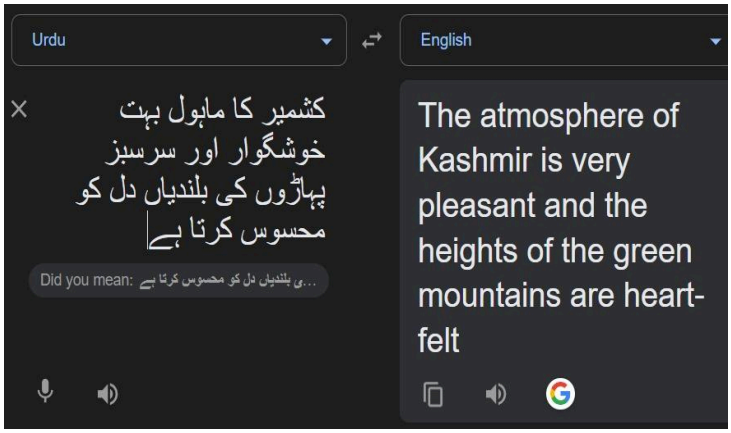
2.




Urdu
English
سیب دنیا بھر میں پسندیدہ پھلوں کی صحت کے لئے بہترین اور لذیذ پھل
Apples are the world's favorite fruit for health and deliciousness

3.


Urdu
English
"کشمیر دنیا کے خوبصورت ترین خطوں میں سے ایک ہے۔ فطرت کی خوبصورتی اور پہاڑوں کی بلندیاں دم توڑ دیتی ہیں۔ کشمیر کی آب و ہوا بہت خوشگوار اور پر سکون ہے، جو اسے ایک خوبصورت جگہ بناتی ہے۔ یہاں کے باغات، جگمگاتی جھیلیں اور سرسبز پہاڑ اسے جنت بنا دیتے ہیں۔"
"Kashmir is one of the most beautiful regions in the world. The beauty of nature and the heights of the mountains are breathtaking. The climate of Kashmir is very pleasant and calm, which makes it a beautiful place. The gardens, sparkling lakes here. And the lush mountains make it a paradise."


Urdu
English
کشمیر کا ماحول بہت خوشگوار اور سرسبز پہاڑوں کی بلندیاں دل کو محسوس کرتا ہے
Did you mean:
The atmosphere of Kashmir is very pleasant and the heights of the green mountains are heart-felt

References:

[1] Mirfan899. (n.d.). Urdu. Retrieved June 1, 2024, from https://mirfan899.github.io/Urdu/

[2] Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., ... & Raffel, C. (2020). mT5: A massively multilingual pre-trained text-to-text transformer. arXiv preprint arXiv:2010.11934.

[3] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research, 21(140), 1-67.

[4] Google. (n.d.). google/mt5-small. Hugging Face. Retrieved June 1, 2024, from https://huggingface.co/google/mt5-small