Marathi Language Question Answering With Transformers

CONVERSATIONAL AI: NATURAL LANGUAGE PROCESSING

Submitted by:

Anchal (102103227)

Yashvardhan (102117217)

BE, Third Year

Submitted to:

Mr. Jasmeet Singh

(Lecturer)

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science and Engineering Department Thapar Institute of Engineering and Technology, Patiala

June 2024

INDEX

S.NO	CONTENT	PAGE NO
1.	Description of NLP Application and Dataset	2
2.	Description of Transformer Model used	4
3.	Python code (Jupyter Notebook/colab Notebook complete)	9

Required Links

1.ColabNotebook

https://colab.research.google.com/drive/1hHrnqU3bm541Oy B9sZ-E3UH30LSdqmeh#scrollTo=01GZT2H3Yjpy&uniqifier=1

2.Figures Canva Link

https://www.canva.com/design/DAGG3pM1rTQ/OknBYMU9gd HrrwBH1gA8BA/edit?utm_content=DAGG3pM1rTQ&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Description of NLP Application and Dataset

1.1 Overview

Marathi question answering (QA) involves developing systems that can understand questions posed in Marathi and provide accurate answers based on a given context. Leveraging transformer-based models, specifically designed for natural language processing (NLP) tasks, significantly enhances the performance of such systems. This application is particularly useful in educational platforms, customer service automation, and information retrieval systems catering to Marathi-speaking users.

1.2 Key Components:-

1.2.1 Dataset :

Dataset used - amitagh/marathi-orca-v05

Link for dataset -https://huggingface.co/datasets/amitagh/marathi-orca-v05

The amitagh/marathi-orca-v05 dataset is designed for developing and evaluating question-answering systems specifically for the Marathi language. The dataset consists of 100,000 rows and is structured to facilitate the training and fine-tuning of machine learning models, particularly transformer-based models, for tasks involving natural language understanding and response generation in Marathi.

Features:-

- row_num
- Id
- eng_system_prompt
- mar_system_prompt
- eng_question
- mar_question
- eng_response
- Mar_response

Conclusion

The amitagh/marathi-orca-v05 dataset is a comprehensive resource for developing sophisticated question-answering systems in Marathi. It supports the creation of models that can understand, interpret, and generate accurate responses to questions, contributing to advancements in NLP for regional languages.

1.3 Model Selection:

1.3.1 Pre-trained Model- MahaBERT-SQuAD

MahaBERT-SQuAD is a MahaBERT model fine-tuned on the translated Marathi question-answering dataset L3Cube-MahaSQuAD

Link- https://huggingface.co/l3cube-pune/marathi-question-answering-squad-bert

1.4 Inference Pipeline:

Question Processing: Tokenize and preprocess the input question.

Contextual Understanding: Use the fine-tuned transformer model to understand the context and predict the answer.

Answer Extraction: Extract the predicted answer span from the context and present it as the final answer.

Description of Transformer Model used

2.1 Pipelines

In BERT transformers, a pipeline refers to a high-level API that allows you to perform various natural language processing tasks easily by providing pre-trained models and tokenizers. For Marathi language, the code you provided loads a Marathi question answering model and utilizes it to answer questions based on the Marathi ORCA dataset.

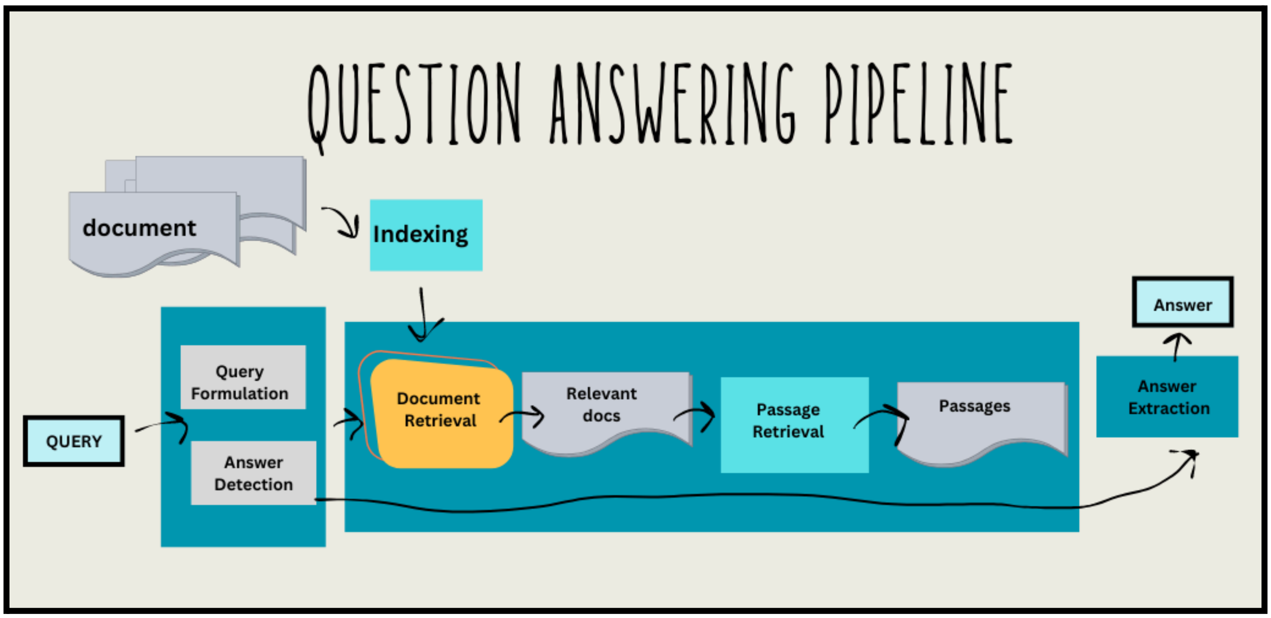


Fig 2.1(question answering pipeline)

The pipelines are a great and easy way to use models for inference. These pipelines are objects that abstract most of the complex code from the library, offering a simple API dedicated to several tasks, including Named Entity Recognition, Masked Language Modeling, Sentiment Analysis, Feature Extraction and Question Answering. See the task summary for examples of use.

There are two categories of pipeline abstractions to be aware about:

The pipeline() which is the most powerful object encapsulating all other pipelines.

Task-specific pipelines are available for audio, computer vision, natural language processing, and multimodal tasks.

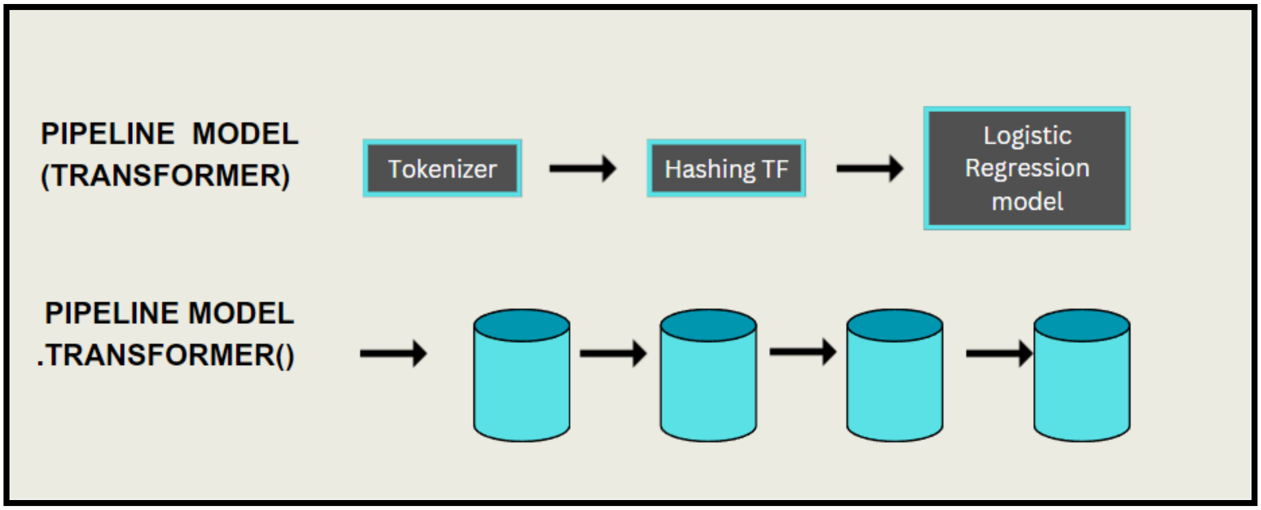


Fig 2.2(pipeline)

2.1.1 The pipeline abstraction

The *pipeline* abstraction is a wrapper around all the other available pipelines. It is instantiated as any other pipeline but can provide additional quality of life

2.1.2 Pipeline batching

All pipelines can use batching. This will work whenever the pipeline uses its streaming ability (so when passing lists or Dataset or generator).

2.2 Question Answering Model

Question-Answering Models are machine or deep learning models that can answer questions given some context, and sometimes without any context (e.g. open-domain QA). They can extract answer phrases from paragraphs, paraphrase the answer generatively, or choose one option out of a list of given options, and so on. It all depends on the dataset it was trained on (e.g. SQuAD, CoQA, etc.) or the problem it was trained for, or to some extent the neural network architecture.

So, for example, if you feed this paragraph (context) to your model trained to extract answer phrases from context, and ask a question like "What is a question-answering model?", it should output the first line of this paragraph.

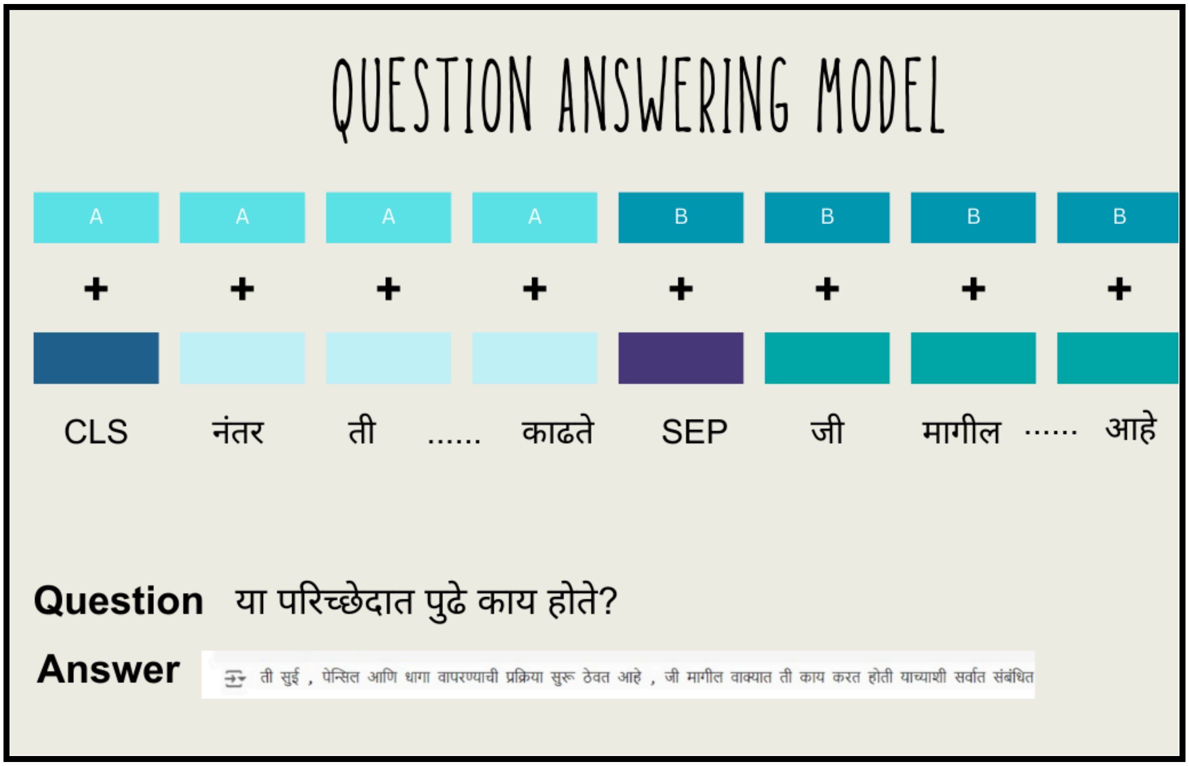


Fig 2.3(question answering Model)

So, for example, if you feed this paragraph (context) to your model trained to extract answer phrases from context, and ask a question like "What is a question-answering model?", it should output the first line of this paragraph.

Such models need to understand the structure of the language, have a semantic understanding of the context and the questions, have an ability to locate the position of an answer phrase, and much more. So without any doubt, it is difficult to train models that perform these tasks. Fortunately, the concept of attention in neural networks has been a lifesaver for such difficult tasks. Since its introduction for sequence modeling tasks, lots of RNN networks with sophisticated attention mechanisms like R-NET, FusionNet, etc. have shown great improvement in QA tasks. However, a completely new neural network architecture based on attention, specifically self-attention, called Transformer, has been the real game-changer in NLP.

For the Question Answering task, BERT takes the input question and passage as a single packed sequence. The input embeddings are the sum of the token embeddings and the segment embeddings. The input is processed in the following way before entering the model:

1.Token embeddings: A **[CLS]** token is added to the input word tokens at the beginning of the question and a **[SEP]** token is inserted at the end of both the question and the paragraph.

2.***Segment embeddings:*** A marker indicating Sentence A or Sentence B is added to each token. This allows the model to distinguish between sentences. In the below example, all tokens marked as A belong to the question, and those marked as B belong to the paragraph.

2.3.1 BLEU(Bilingual Evaluation Understudy)

BLEU (Bilingual Evaluation Understudy) is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. Quality is considered to be the correspondence between a machine's output and that of a human: "the closer a machine translation is to a professional human translation, the better it is" – this is the central idea behind BLEU. BLEU was one of the first metrics to claim a high correlation with human judgements of quality, and remains one of the most popular automated and inexpensive metrics.

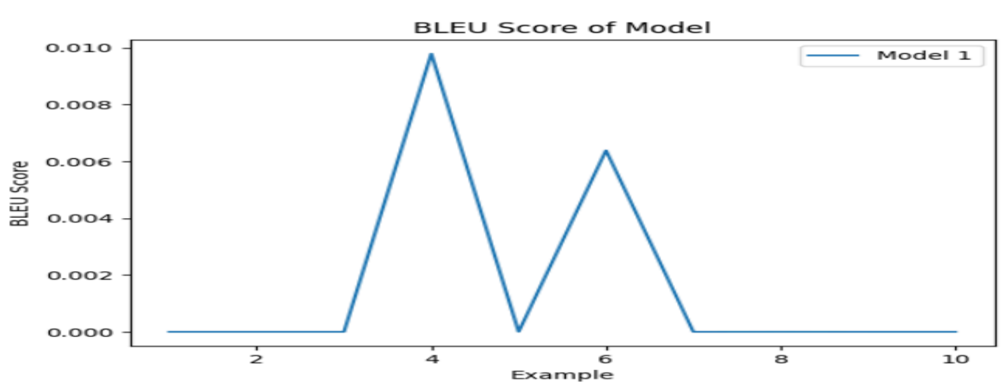


Fig 2.3(Bilingual Evaluation Understudy)

Scores are calculated for individual translated segments—generally sentences—by comparing them with a set of good quality reference translations. Those scores are then averaged over the whole corpus to reach an estimate of the translation's overall quality. Neither intelligibility nor grammatical correctness are not taken into account.

2.3.2 Metric Description

BLEU (Bilingual Evaluation Understudy) is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. Quality is considered to be the correspondence between a machine's output and that of a human: "the closer a machine translation is to a professional human translation, the better it is" – this is the central idea behind BLEU. BLEU was one of the first metrics to claim a high correlation with human judgements of quality, and remains one of the most popular automated and inexpensive metrics.

Scores are calculated for individual translated segments—generally sentences—by comparing them with a set of good quality reference translations.

Python code (colab Notebook)

3.1 Link for Colab Notebook

😀https://colab.research.google.com/drive/1hHrnqU3bm541OyB9sZ-E3UH30LSdqmeh#scrollTo=01GZT2H3Yjpy&uniqifier=1

Complete Notebook is given below

```
!pip install transformers datasets torch

from transformers import pipeline

from datasets import load_dataset

# Load the Marathi question answering model

qa_pipeline = pipeline(

    "question-answering",

    model="l3cube-pune/marathi-question-answering-squad-bert",

    tokenizer="l3cube-pune/marathi-question-answering-squad-bert"

)

# Load the Marathi ORCA dataset

dataset = load_dataset("amitagh/marathi-orca-v05")



# Iterate through the dataset and perform question answering for first 10 rows

for i in range(10):

  example=dataset['train'][i]

  context = example["mar_response"]
```

```python
question = example["mar_question"]

  # Perform question answering

result = qa_pipeline(question=question, context=context)

print('Example:',i+1)

print("Question:", question)

print("Context:", context)

print("Answer:", result["answer"])

print("----------------")
```

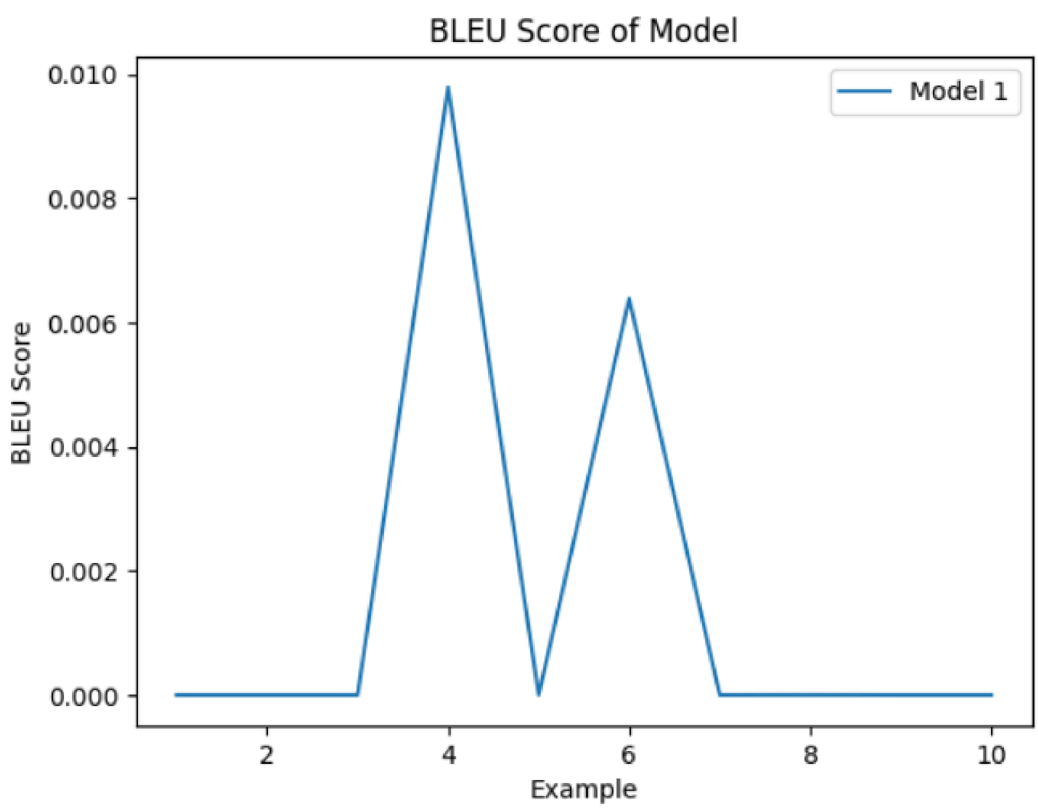
```
Example: 1
Question: या सर्व डेटाचे वर्णन करणारे अंदाजे पंधरा शब्दांचे वाक्य तयार करा: मिडसमर हाऊस eatType restaurant; मिडसमर हाउस फूड चायनीज; मिडसमर हाऊस किंमत श्रेणी मध्यम; मिडसमर हाऊस ग्राहक रेटिंग 5 पैकी 3; ऑल बार वन जवळ मिडसमर
Context: मिडसमर हाऊस हे 3/5 ग्राहक रेटिंग असलेले माफक किमतीचे चीनी रेस्टॉरंट आहे, जे ऑल बार वन जवळ आहे.
Answer: 3/5
----------------
Example: 2
Question: या परिच्छेदात पुढे काय होते?

मग ती कापसाच्या बॉलवर सुई घासते आणि मग ती पेन्सिलवर ढकलते आणि त्याभोवती धागा गुंडाळते. ती नंतर उत्पादनाचा एक बॉक्स धरते आणि नंतर एका वाडग्यात अनेक द्रव ओतते. ती
तुमचे उत्तर यामधून निवडा: A. सॉसपॅन जोडते आणि ग्राइंडरमध्ये उत्पादन हलवते. बी. सिगारेट स्टाईल करण्यासाठी धागा चिमटा काढतो आणि मग निघून जातो. C. नंतर सुईला शाईत बुडवतो आणि पेन्सिलचा वापर करून तिच्या पायावर डिझाईन काढतो, शेवटी चिंधीने
Context: C. नंतर ती सुई शाईत बुडवते आणि पेन्सिलचा वापर करून तिच्या पायावर एक डिझाईन काढते, शेवटी चिंधीने घासते. या पर्यायामध्ये, ती सुई, पेन्सिल आणि धागा वापरण्याची प्रक्रिया सुरू ठेवत आहे, जी मागील वाक्यात ती काय करत होती याच्याशी सर्वात
Answer: डिझाईन
----------------
Example: 3
Question: कृपया खालील प्रश्नाचे उत्तर द्या: मला विद्यार्थ्यांची परिच्छेद वाचण्याची आणि त्याबद्दलच्या प्रश्नांची उत्तरे देण्याची क्षमता तपासायची आहे. तुम्ही कृपया उताऱ्यासाठी एक चांगला प्रश्न विचारू शकता का "1901 मध्ये, फेडरेशन ऑफ ऑस्ट्रेलिया ही प्रक्रिया होती ज्या
उत्तर:
Context: उताऱ्यावर आधारित, 1901 फेडरेशन ऑफ ऑस्ट्रेलियाच्या प्राथमिक प्रेरणा आणि परिणामांची चर्चा करा, ज्यात संघराज्य सरकारच्या भूमिका आणि जबाबदाऱ्या, तसेच सहभागी असलेल्या वैयक्तिक राज्यांच्या सतत सरकारी संरचनांचा समावेश आहे.
Answer: 1901 फेडरेशन ऑफ ऑस्ट्रेलियाच्या
----------------
Example: 4
Question: जेम्स एक टीव्ही शो चालवतो आणि त्यात 5 मुख्य पात्रे आणि 4 लहान पात्रे आहेत. तो लहान पात्रांना प्रत्येक भागासाठी $15,000 देतो. त्याने प्रमुख पात्रांना तिप्पट पैसे दिले. तो प्रति एपिसोड किती देतो? चला शक्य तितके अचूक असू द्या.
Context: जेम्स लहान पात्रांना प्रत्येक भागासाठी $15,000 देते. 4 किरकोळ वर्ण असल्याने, तो त्यांना एकूण 4 * $15,000 = $60,000 प्रति एपिसोड देतो.

प्रमुख पात्रांना तिप्पट पैसे दिले जातात. तर, प्रत्येक प्रमुख पात्राला 3 * $15,000 = $45,000 प्रति एपिसोड दिले जातात.
```

```python
from nltk.translate.bleu_score import sentence_bleu, SmoothingFunction


def calculate_bleu_score(reference, candidate):
    """
    Calculates BLEU score for a candidate sentence against a reference sentence.

    Args:
        reference (str): The reference sentence.
        candidate (str): The candidate sentence to be evaluated.

    Returns:
        float: The BLEU score of the candidate sentence.
    """
    # Split sentences into tokens (words)
    reference_tokens = reference.split()
    candidate_tokens = candidate.split()

    # Calculate BLEU score using NLTK's sentence_bleu function
    bleu_score = sentence_bleu([reference_tokens], candidate_tokens, smoothing_function=SmoothingFunction().method4)
```

```
    return bleu_score

import matplotlib.pyplot as plt



# ... (rest of the code remains the same)



# Create lists to store the BLEU scores

bleu_scores_model1 = []

bleu_scores_model2 = []



# Iterate through the dataset

for i in range(10):

  example = dataset['train'][i]

  context = example["mar_response"]

  question = example["mar_question"]



  # Perform question answering with both models

  result1 = qa_pipeline(question=question, context=context)

  answer1 = result1["answer"]
```

```
result2 = other_qa_pipeline(question=question, context=context)

answer2 = result2["answer"]



# Calculate BLEU score for each answer against the reference question

bleu_score1 = calculate_bleu_score(question, answer1)

bleu_score2 = calculate_bleu_score(question, answer2)



# Append the BLEU scores to the lists

bleu_scores_model1.append(bleu_score1)

bleu_scores_model2.append(bleu_score2)



print('Example:', i + 1)

print("Question:", question)

print("Context:", context)

print("Answer (Model 1):", answer1)

print("BLEU Score (Model 1):", bleu_score1)

print("Answer (Model 2):", answer2)

print("BLEU Score (Model 2):", bleu_score2)

print("----------------")
```

```
# Create a line graph of the BLEU scores

plt.plot(range(1, 11), bleu_scores_model1, label='Model 1')

plt.plot(range(1, 11), bleu_scores_model2, label='Model 2')

plt.xlabel('Example')

plt.ylabel('BLEU Score')

plt.title('BLEU Scores for Both Models')

plt.legend()

plt.show()
```




```
from transformers import BertForQuestionAnswering
```

```
model=BertForQuestionAnswering.from_pretrained('l3cube-pune/marathi-question-answering-squad-bert')

from transformers import BertTokenizer



tokenizer=BertTokenizer.from_pretrained('l3cube-pune/marathi-question-answering-squad-bert')

question = "या परिच्छेदात पुढे काय होते?"

answer_text = '''नंतर ती सुई शाईत बुडवते आणि पेन्सिलचा वापर करून तिच्या पायावर एक डिझाईन काढते, शेवटी चिंधीने घासते. या पर्यायामध्ये, ती सुई, पेन्सिल आणि धागा वापरण्याची प्रक्रिया सुरू ठेवत आहे, जी मागील वाक्यात ती काय करत होती याच्याशी सर्वात संबंधित आहे.'''

input_ids = tokenizer.encode(question,answer_text)

print(input_ids)

token_to_ids=tokenizer.convert_ids_to_tokens(input_ids)

for token,id in zip(token_to_ids,input_ids):

  print(token,id)

sep_index=input_ids.index(tokenizer.sep_token_id)

num_seg_a=sep_index+1

print(num_seg_a)

num_seg_b=len(input_ids)-num_seg_a

print(num_seg_b)

segment_ids=[0]*num_seg_a+[1]*num_seg_b
```

```
print(segment_ids)

!pip install torch

import torch

outputs=model(torch.tensor([input_ids]),

              token_type_ids=torch.tensor([segment_ids]),

              return_dict=True)

start_probs=outputs.start_logits

end_probs=outputs.end_logits

print(start_probs)



start_index=torch.argmax(start_probs)

end_index=torch.argmax(end_probs)

print(start_index)

answer=' '.join(token_to_ids[start_index:end_index+1])

print(answer)

answer=token_to_ids[start_index]

for i  in range(start_index+1,end_index+1):

  if token_to_ids[i][0:2]=='##':

    answer+=token_to_ids[i][2:]

  else:
```

```
    answer+=' '+token_to_ids[i]

print(answer)
```

```
[20] answer=' '.join(token_to_ids[start_index:end_index+1])

[21] print(answer)

ती सुई , पेन ##्सिल आणि धागा वापर ##ण्याची प्रक्रिया सुरू ठेवत आहे , जी मागील वाक्य ##ात ती काय करत होती याच्या ##शी सर्वात संबंधित

[22] answer=token_to_ids[start_index]
     for i  in range(start_index+1,end_index+1):
       if token_to_ids[i][0:2]=='##':
         answer+=token_to_ids[i][2:]
       else:
         answer+=' '+token_to_ids[i]
     print(answer)

ती सुई , पेन्सिल आणि धागा वापरण्याची प्रक्रिया सुरू ठेवत आहे , जी मागील वाक्यात ती काय करत होती याच्याशी सर्वात संबंधित
```

```
labels=[]

for (ids,token) in enumerate(token_to_ids):

   labels.append('{:} - {:>2}'.format(token, ids))

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(16,16))

s=start_probs.detach().numpy().flatten()

ax=sns.barplot(x=token_to_ids,y=s,ci=None)

ax.set_xticklabels(ax.get_xticklabels(), rotation=90, ha="center")

ax.grid(True)
```

```
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: Us
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: Us
  fig.canvas.print_figure(bytes_io, **kw)
```