Question Answering and Entity Recognition Using Natural Language Processing

INTRODUCTION

Question Answering Using Bert

Numerous machine learning algorithms attempt to answer questions using natural language. The bag of words was popular in the past, and it attempted to answer questions that were pre-defined by the developers. Developers must spend a significant amount of time crafting questions and answers for each question when using this strategy. This strategy worked well for chatbots, but it could not answer questions for a large database. Models known as transformers dominate modern natural language processors. Many machine learning algorithms aim to answer questions in natural language. In the past, the bag of words was popular, and it attempted to answer questions that the developers had pre-defined. When employing this method, developers must devote a significant amount of time to writing each question's questions and answers. This method worked well for chatbots, but it could not handle a vast database of queries. Modern natural language processors rely heavily on transformer models. Question answering has always been an important aspect of any language study. The word "question answering" stems from a reading comprehension exercise in which the reader is given a set of paragraphs to read and answers questions about them. Many models are trained to do so using machine learning methods based on this principle. They can comprehend the context of the material and respond to questions about it. They can respond in the natural language we are accustomed to. It not only provides replies but also attempts to comprehend the sentiments of inquiries and locate the text or sentence in question. To do so, I will be using the "Bert Model from HugginFace Transformers," which can be downloaded and used on the HugginFace websites. I will also be using other Python libraries to accomplish this.

Entity Recognition Using Spacy

Now for the second phase, I have done entity recognition, for that, I have used Spacy which is a Python library for advanced Natural Language Processing (NLP). It is free and open-source. If you work with a lot of text, you will want to learn more about it eventually. What is it about, for example? In what context do the terms mean? What exactly is being done to whom? What products and firms are mentioned? Which texts resemble one another? Spacy is a production-ready tool that allows you to create apps that analyze and "understand" massive amounts of text. It can be used to create data extraction and natural language understanding systems, as well as to pre-process text for deep learning. I have used different NER models which is a common natural language processing (NLP) activity that recognizes and automatically identifies preset entities in a text. Person names, organizations, dates and times, and locations are all significant pieces of information to extract from unstructured, unlabeled raw text. By the end of this report, you'll be able to use HuggingFace Transformers and SpaCy in Python to perform named entity recognition on any English text.

SQUAD

SQUAD stands for Stanford Question Answering Dataset, and it is a reading comprehension dataset. It's a set of questions on Wikipedia articles that are answered by crowd workers, with each question's response being a text segment or span from the relevant reading section. SQuAD 1.1 was used to fine-tune the model, which has 100,000+ question-answer pairs on 500+ pages. I have selected a paragraph on Rabindranath Tagore from Wikipedia to show both the models Question-Answering as well as Entity Recognition. Here is the link - https://en.wikipedia.org/wiki/Rabindranath_Tagore

METHODOLOGY (Question Answering)

In the first part, I have started with answering questions, here are the details explained in brief.

Processing of Input

We employ Bert-model(bert-large-uncased-wholeword-masking-finetuned-squad) for this experiment, which has previously been pre-trained and fine-tuned on the SQUAD V1 dataset, which has 100,000+ questions. Bert treats the input question and passage (context) as a single packed sequence for the question-answering task. The input embeddings are made up of the token and segment embeddings added together. The input processing procedure is shown below.

Token embeddings: At the start of the question, a [CLS] token is added to the input word tokens, and a [SEP] token is inserted at the end of both the question and the paragraph. Segment embeddings: Each token has a tag that indicates whether it belongs to Sentence A or Sentence B. This enables the model to recognize different sentences. All tokens marked with an A belong to the question, whereas those marked with a B belong to the paragraph in the example below. Here's an example of input processing with passages, questions, tokens, segment ids, and input ids. Note that the segment ids only comprise 0 and 1 which denotes query and a passage\stext correspondingly.

```
"""
question: What is the name of YouTube Channel
passage: Watch complete playlist of Ishika Nisha. Don't forget to like, share and subscribe my channel IN Tech Team
"""
#Tokenize input question and passage
#Add special tokens - [CLS] and [SEP]
input_ids = tokenizer_for_bert.encode (question, passage,  max_length= max_len, truncation=True)
"""
[101, 2054, 2003, 1996, 2171, 1997, 7858, 3149, 102, 3422, 3143, 2377, 9863, 1997, 3019, 2653, 6364, 1012,
2123, 1005, 1056, 5293, 2000, 2066, 1010, 3745, 1998, 4942, 29234, 2026, 3149, 1045, 2290, 6627, 2136, 102]
"""
#Getting number of tokens in 1st sentence (question) and 2nd sentence (passage that contains answer)
sep_index = input_ids.index(102)
len_question = sep_index + 1
len_passage = len(input_ids)- len_question
"""
8
9
27
"""
#Need to separate question and passage
#Segment ids will be 0 for question and 1 for passage
segment_ids =  [0]*len_question + [1]*(len_passage)
"""
[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]
"""
#Converting token ids to tokens
tokens = tokenizer_for_bert.convert_ids_to_tokens(input_ids)
"""
tokens = ['[CLS]', 'what', 'is', 'the', 'name', 'of', 'youtube', 'channel', '[SEP]', 'watch', 'complete',
'play', '##list', 'of', 'ishika', 'nisha', '.', 'don', "'", 't', 'forget', 'to', 'like',
',', 'share', 'and', 'sub', '##scribe', 'my', 'channel', 'i', '##n', 'tech', 'team', '[SEP]']
"""
#Getting start and end scores for answer
```

Fig. 1

Obtaining an Answer

It constructs a word embedding and introduces a start vector after passing the input tokens through layers of transformers. The likelihood of each word being the start word is computed by taking the dot product of the word's final embedding and the start vector, then applying SoftMax to all of the words. The starting point for the answer is the word with the highest probability value.

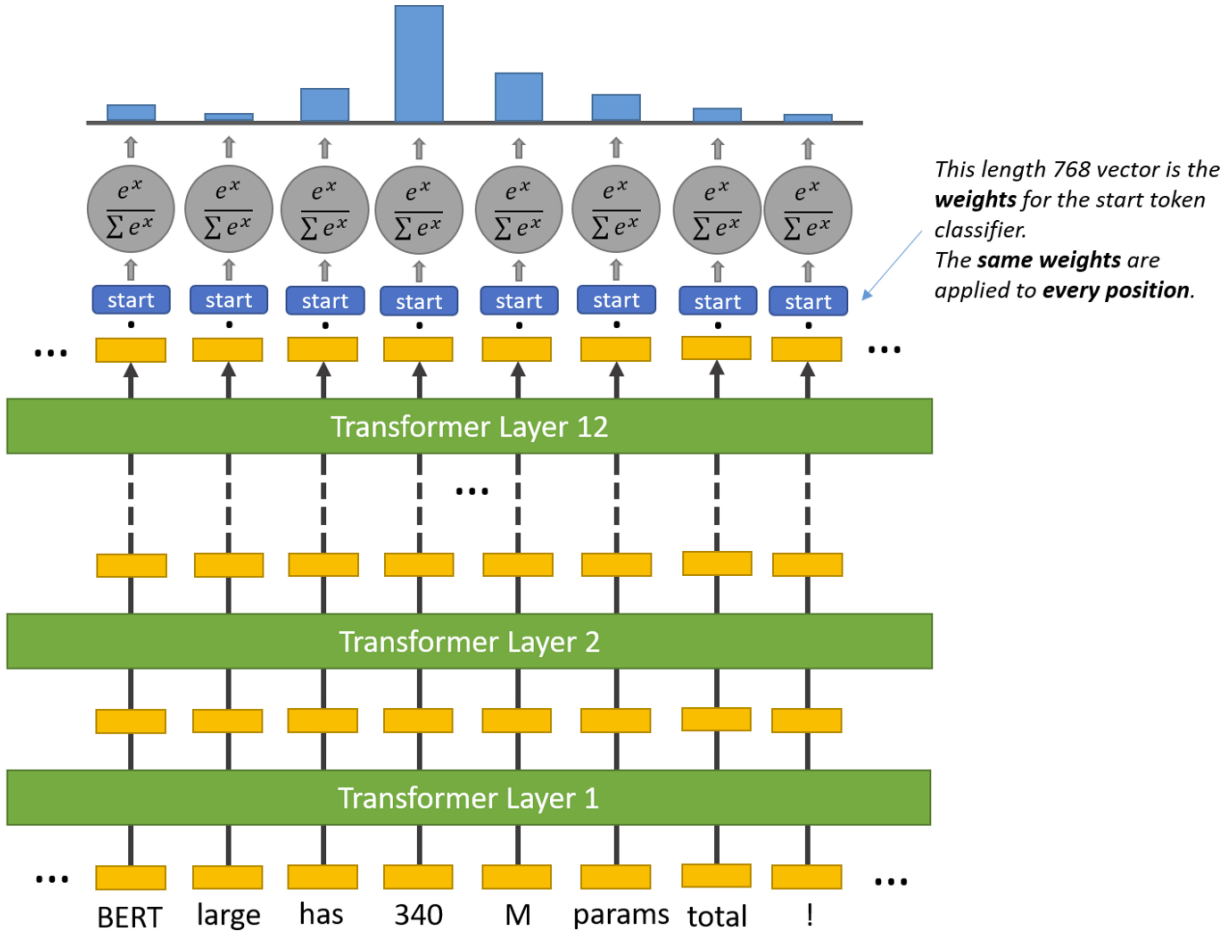

Fig. 2

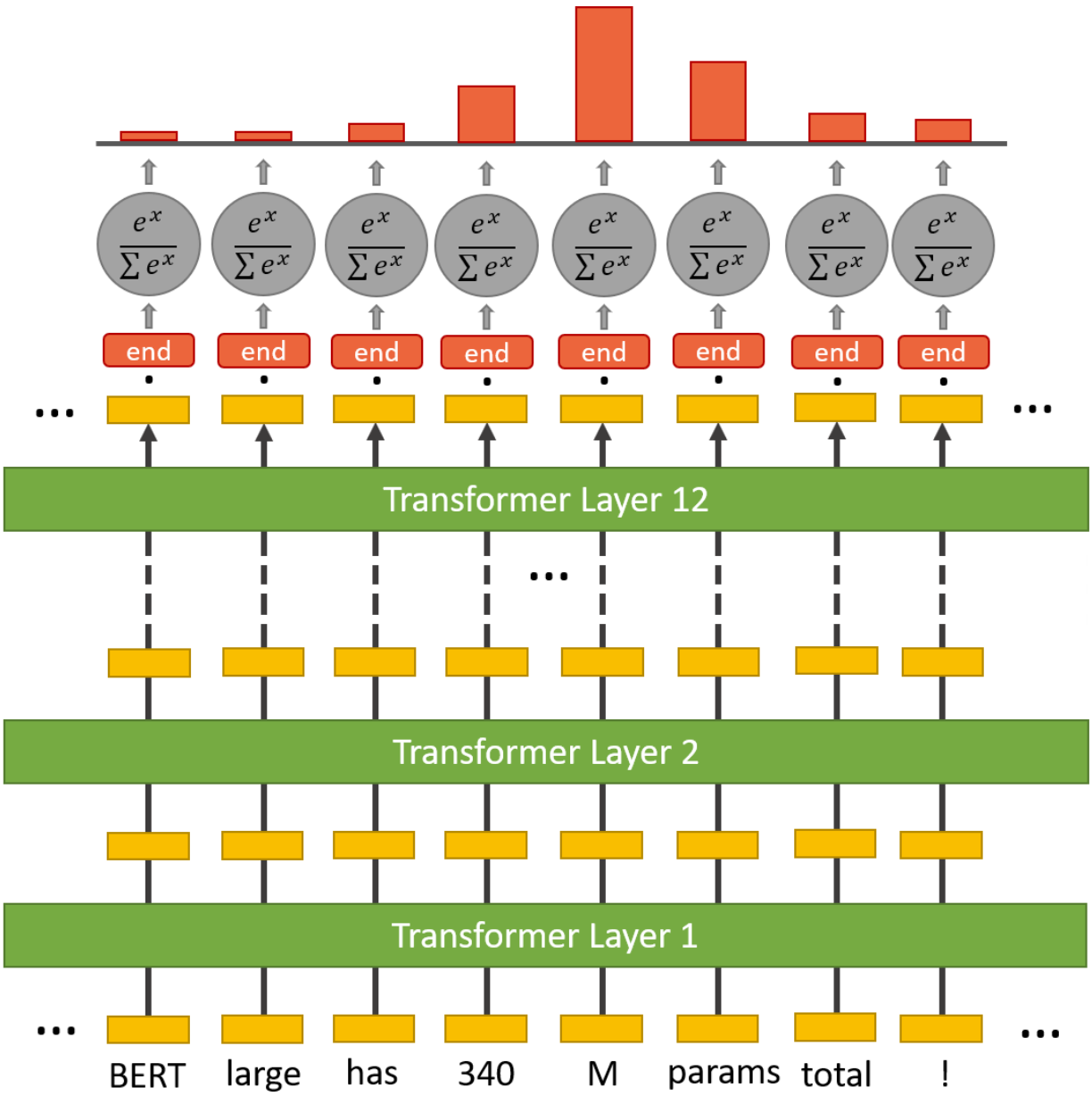
$\frac{e^x}{\sum e^x}$
end
Transformer Layer 12
Transformer Layer 2
Transformer Layer 1
BERT
large
has
340
M
params
total
!

Fig. 3

Setting up the Environment

This will install all of the project's needed packages. The heart of this project is made up of two main python modules that I wrote.

1. loadModel.py is a Python script that loads a model. This module will load the BERT model, which will then process the inputs and produce outputs. We'll utilize the QAPipe class in this module to interface with the BERT model. We will also use the pre-train bert model so that the fresh connected layer can learn from it.

```
from transformers import BertForQuestionAnswering
from transformers import BertTokenizer
import torch
import numpy as np

[6] #Step 3: Load pre-trained Bert model
model = BertForQuestionAnswering.from_pretrained('bert-large-uncased-whole-word-masking-finetuned-squad')

tokenizer_for_bert = BertTokenizer.from_pretrained('bert-large-uncased-whole-word-masking-finetuned-squad')
```

Fig. 4

2. Input_ids.py: This module will create a dataset from either text or article from a web address. It has a class named Input_ids as shown in fig 1, which is used to generate the clean context from either a text passage or article on the internet.

The Process (Question Answering)

I have made a process chart so that it's easy to understand, and everyone can have a better insight

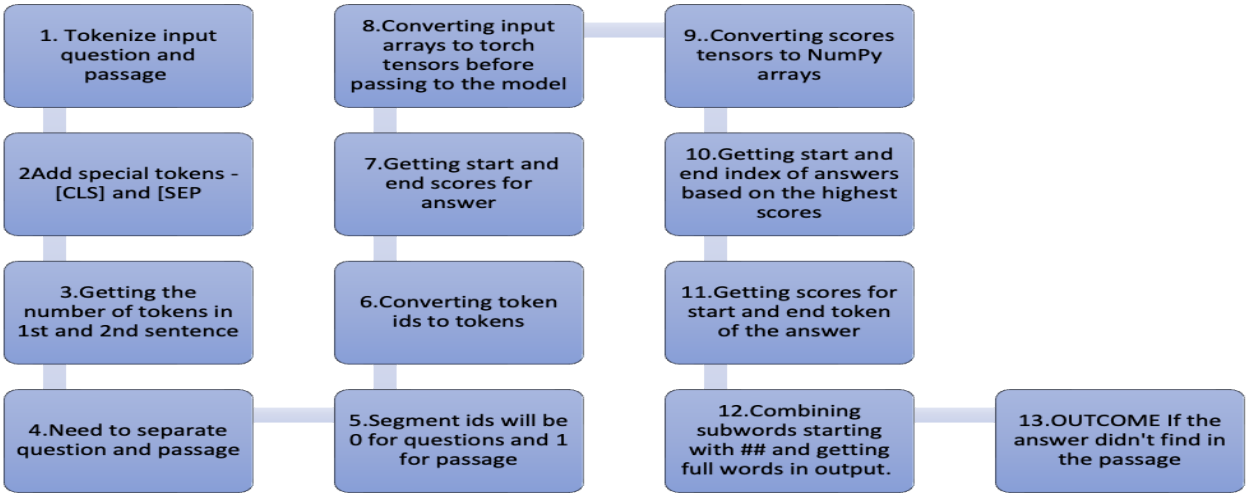

Fig. 5

VISUALIZATION

Below, in Fig 6, I have shown the visualization, of the final output

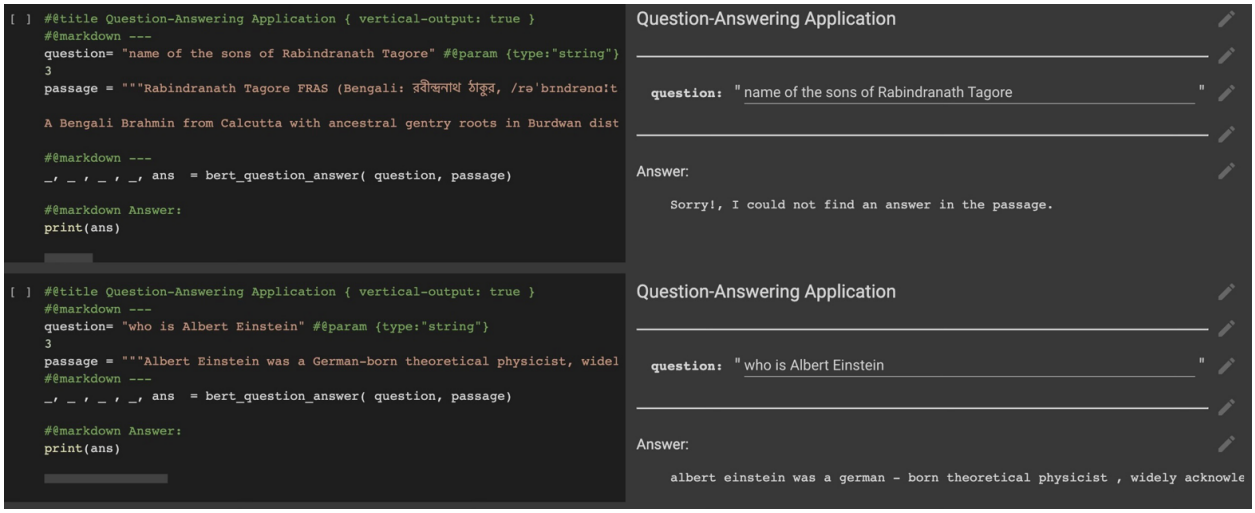

Fig. 6

Note, if something is not presented in the output, it returns the answer as Sorry, could not find the answer in the passage.

METHODOLOGY (Entity Recognition)

Our purpose is to evaluate NER technologies that are freely available and perform well for our research projects. The following are the criteria we used to make our decision:

a) The NER tool is open source and can be used indefinitely.

b) The tool can be downloaded and installed locally and is functional with the default settings.

c) The tool hasn't been trained for a certain domain.

d) The tool must be able to distinguish between the three primary entity types: person, location, and organization.

e) Select the best possible tool.

The Process

There are a lot of other models that were fine-tuned on the same dataset. Here are a few models used

(NER-TRANSFORMERS)

1.dslim/bert-base-NER

The named entities of this dataset are:

- O: Outside of a named entity.
- B-MIS: Beginning of a miscellaneous entity right after another miscellaneous entity.
- I-MIS: Miscellaneous entity.
- B-PER: Beginning of a person's name right after another person's name.
- I-PER: Person's name.
- B-ORG: The beginning of an organization right after another organization.
- I-ORG: Organization.
- B-LOC: Beginning of a location right after another location.
- I-LOC: Location.

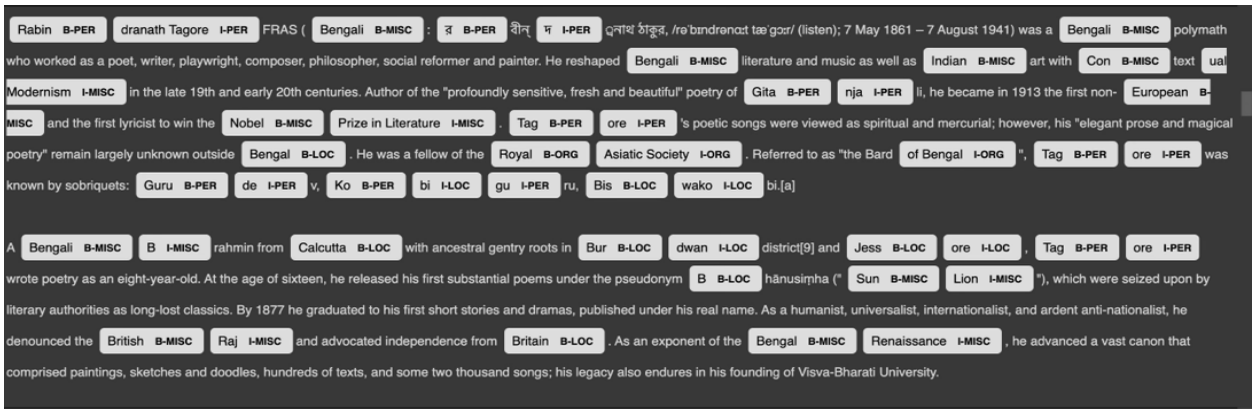

Fig. 7

2.xlm-roberta-large-finetuned-conll03-english

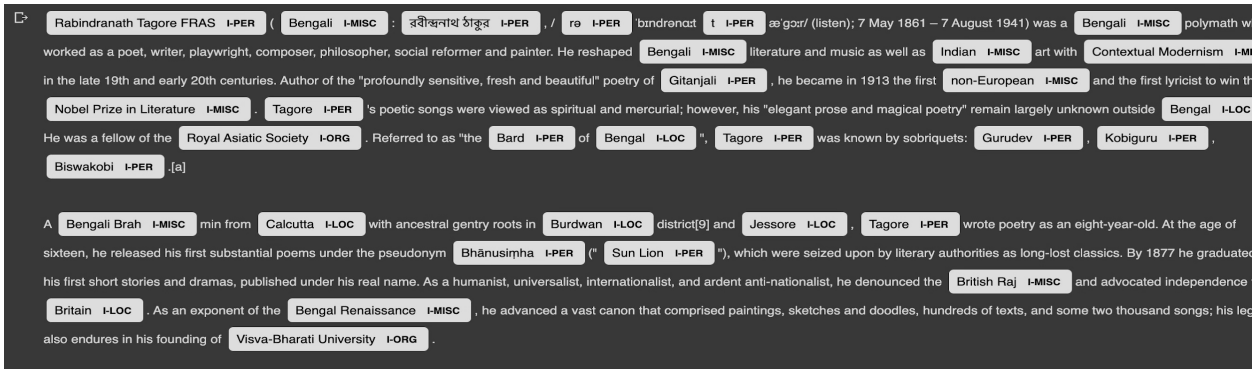

Fig. 8

As you can see it performs slightly better as it describes Rabindranath Tagore as a single entity and also the district Jessore.

3.roJean-Baptiste/roberta-large-ner-english”

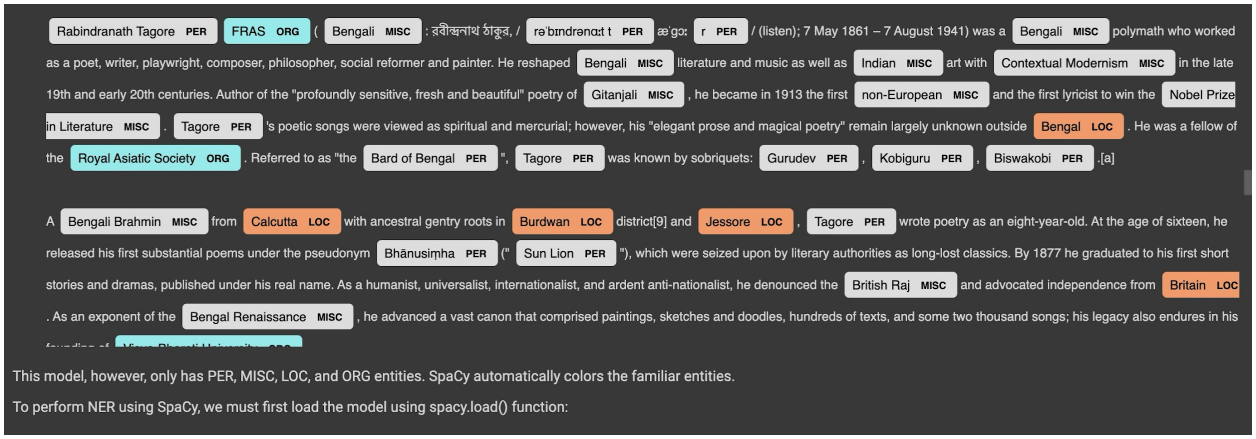

Fig. 9

Pipeline Used (SpaCy)

1. "en_core_web_sm” -NER

LANGUAGE	ENEnglish
TYPE	COREVocabulary, syntax, entities
GENRE	WEBwritten text (blogs, news, comments)
SIZE	SM12 MB
COMPONENTS	tok2vec, tagger, parser, senter, attribute_ruler, lemmatizer, ner
PIPELINE	tok2vec, tagger, parser, attribute_ruler, lemmatizer, ner
VECTORS	0 keys, 0 unique vectors (0 dimensions)

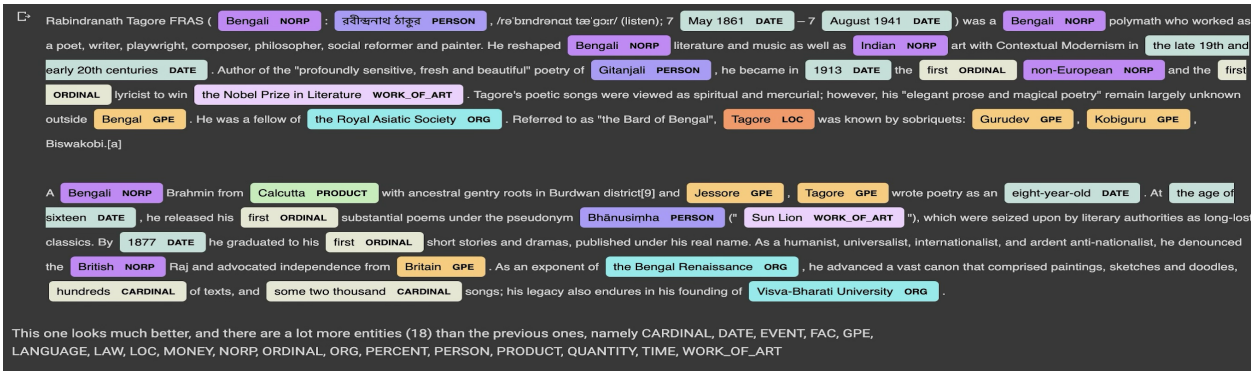

Fig. 10

This one has a lot more entities (12) than the last one, including CARDINAL, DATE, EVENT, FAC, GPE, LANGUAGE, LOC, MONEY, NORP, ORDINAL, ORG, PERSON, TIME, and WORK OF ART.

However, Rabindranath Tagore was not labeled, and in certain places, Tagore was mislabeled as a LOC, thus let's apply spaCy's Transformer model

2. 'en_core_web_trf' – ROBERTA BASE USING SPACY

English transformer pipeline (roberta-base). Components: transformer, tagger, parser, ner, attribute_ruler, lemmatizer.

LANGUAGE	ENEnglish
TYPE	COREVocabulary, syntax, entities
GENRE	WEBwritten text (blogs, news, comments)
SIZE	TRF438 MB
COMPONENTS	transformer, tagger, parser, attribute_ruler, lemmatizer, ner
PIPELINE	transformer, tagger, parser, attribute_ruler, lemmatizer, ner
VECTORS	0 keys, 0 unique vectors (0 dimensions)

This time Rabindranath Tagore was labeled as a person, and Burdhwan is being offered as LOC Hence we can conclude the en_core_web_trf model performs much better than the previous ones. Similar Entities have been colored on the same color scale.

Fig. 11

Results (Insights)

I have made a comparison of both transformer and spacy models so that it's easy for everyone to learn and see the difference. Also below is a table that confirms my findings.

The en_core_web_trf model performs much better than the previous ones. Check this table that shows each English model offered by spaCy with their size and metrics evaluation of each:

Model Name	Model Size	Precision	Recall	F-Score
en_core_web_sm	13MB	0.85	0.84	0.84
en_core_web_md	43MB	0.85	0.84	0.85
en_core_web_lg	741MB	0.86	0.85	0.85
en_core_web_trf	438MB	**0.90**	**0.90**	**0.90**

Table 1

For other languages, spaCy strives to make these models available for every language globally.

CONCLUSION

In this Report, I used the same dataset as described in SQUAD to show how to implement a Wikipedia Named Entity Recognition (NER)method using SPACY, as well as the Question answering method using BERT.

When it comes to entity recognition I wanted to compare all the models and select the best possible one, So in this method, first, a set of entities and types was identified, then different types of models of both spacy and transformer were used, and a different comparison was made on the basis of visualization, entities identification, recall, precision and F-score which can be clearly seen in table 1. The results demonstrated that this new NER method is effective in recognizing Wikipedia-named entities.

When it comes to BERT, it can grasp the language structure and handle dependencies between sentences, in addition to question responses. It can answer queries using simple logic. BERT was able to answer the query "Who is Albert Einstein?", Albert Einstein was a german-born theoretical physicist, widely acknowledged to be one of the greatest and most influential physicists of all time, while the longer passage can be passed through the model, the code will automatically truncate the extra part if the length of the question and passage exceeds 5000 tokens. Also, we can see in the second part that if something is not presented in the output, it returns the answer as Sorry, could not find the answer in the passage. (Fig 6)

For, a detailed reference you can go to my Github - https://github.com/ishikanisha28/INISHA_ADVANCE-MACHINE-LEARNING

FUTURE PERSPECTIVE

1. Named entity recognition (NER) allows you to quickly recognize significant aspects in a document, such as people's names, places, brands, and monetary values. When dealing with enormous datasets, extracting the major entities in a text can help sort unstructured data and uncover relevant information. What do the words mean in context? Who is doing what to whom? What companies and products are mentioned? Which texts are similar to each other?
2. The question answering model can have various uses in the future. One such use is a visual question and answering where this model can be used to answer questions based on a picture or a scanned copy of a written document. This can be extremely helpful for visually impaired people
3. Also, this can be used in the medical field to answer any questions people have related to any disease. For example, during Covid, this model was used to help people answer any questions related to COVID-19 and clear any misconceptions or questions they had. In the future as well this can be a very useful tool to help people know about any disease.

REFERENCES

[1] Horev, Rani. “BERT Explained: State of the Art Language Model for NLP.” Medium. Towards Data Science, November 17, 2018. https://towardsdatascience.com/bertexplained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

[2] “Pretrained Models¶.” Pretrained models - transformers 4.2.0 documentation. Accessed February 3, 2021. https://huggingface.co/transformers/pretrained_models.html.

[3] “SpaCy · Industrial-Strength Natural Language Processing in Python.”Industrialstrength Natural Language Processing in Python. Accessed February 3, 2021. https://spacy.io/.

[4] https://www.thepythoncode.com/article/named-entity-recognition-using-transformers