



1



0



0



0



0

Following

Share

...

Question and Answer using Pre-Trained Transformers

Created 1 mon ago

57 Views

0 Comments

BYOB-2.0



Divya P
@DivyaPalanisamy

0
Followers

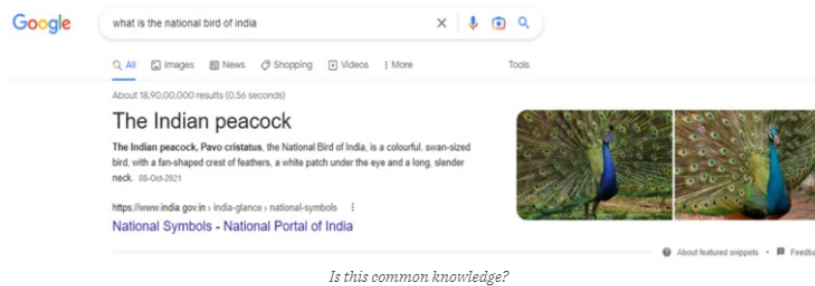
0
Following

2
Posts

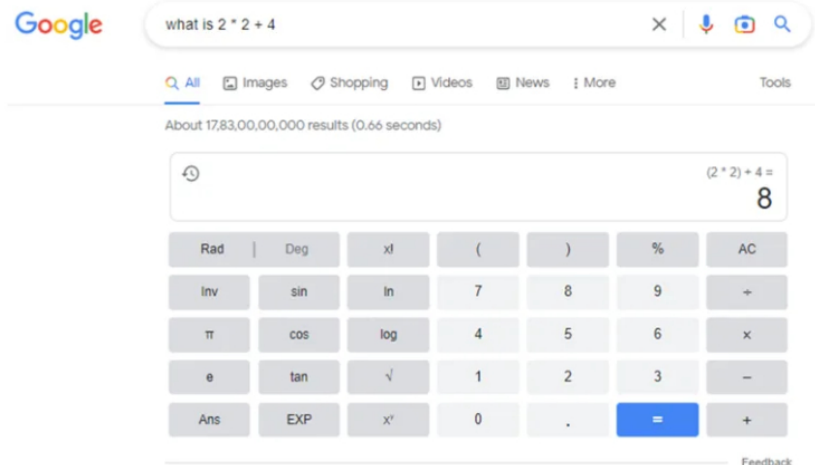
Powered by Habitate

It was always science fiction to pose a question to a machine and receive an answer. Since things have changed, we now use Q&A systems everywhere without even realizing it.

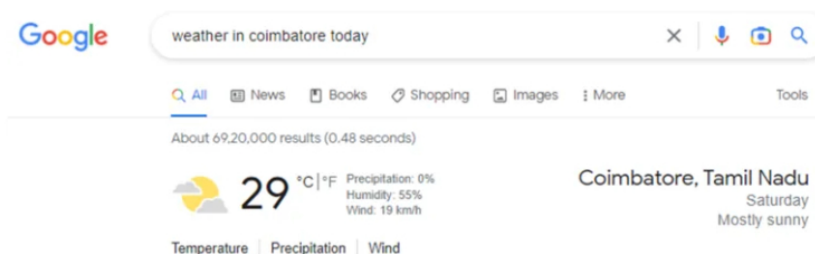
Google searches are the best example. In spite of the fact that Google is often used to obtain information and will only send you on the appropriate path, there are many other queries that are straightforward questions for which Google frequently offers straightforward replies.



Is this common knowledge?



Google identifies math questions quickly.





We can even get more specific and request information based on the location and time.

Google is definitely way ahead of the game. However, that doesn't mean we can't create excellent Q&A systems. We can also use the same models that Google uses in its search.

In this blog, we will understand how to use transformer models for Q&A. We'll look at Google's Bert and show you how to use it. In summary, we will discuss the following:

- HuggingFace's Transformers
 - Installation
- Setting up a Q&A Transformer
 - Finding a Model
 - The Q&A Pipeline
 - Model and tokenizer initialization
 - Tokenization
 - Pipeline and Prediction

HuggingFace's Transformers:

First and foremost, modern NLP is dominated by these incredible models, known as transformers. These models are brilliant, and they are a relatively new development.

Second, the real-world implementation of transformers is almost entirely carried out using a library called transformers, created by an incredible group of people known as HuggingFace.

For a variety of reasons, the Transformers library is popular. But three things in particular immediately come to mind:

Open-source: The library is open-source, with a vibrant community.

Ease of use: With just five lines of code, we can start using models from Google, OpenAI, and Facebook.

Models: The library has an unbelievably massive collection of models that can be downloaded and used along with installation.

Installation:

```
!pip install transformers
```

However, the library requires PyTorch or TensorFlow to be installed as well, because these are used in the background.

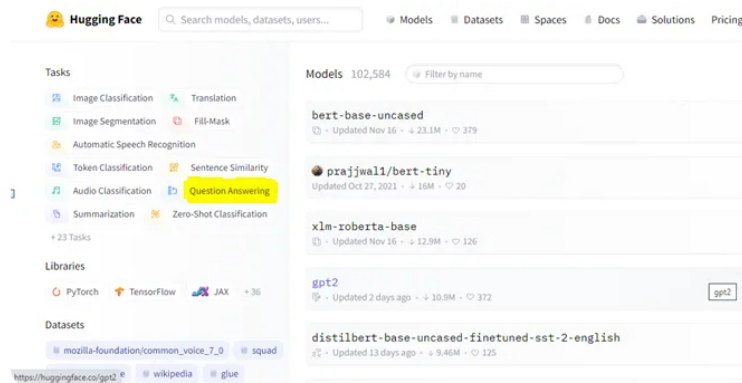
Setting up a Q&A Transformer:

Finding a Model:

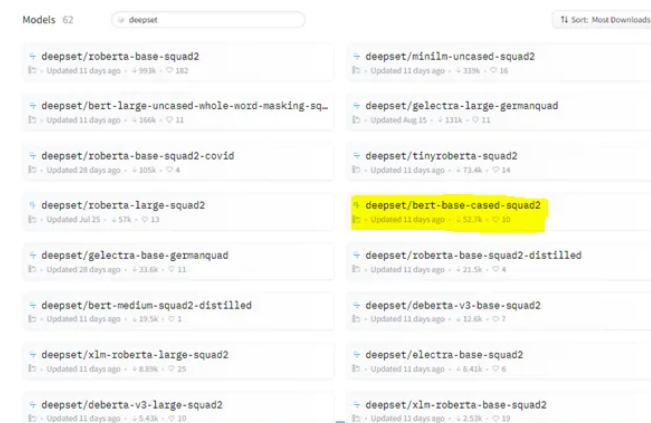
Transformers are installed, so we can begin to build our first Q&A transformer script.

Let's start by searching for a model to use. We simply go to huggingface.co/models and

Let's start by searching for a model to use. We simply go to huggingface.co/models and then select Question-Answering from the menu on the left.



We can also search for certain models. In this instance, I'm implementing the "deepset/bert-base-cased-squad2" model.



Deepset constructed this model. AI is the reason for the deepset/. They were pre-trained on the SQuAD 2.0 dataset for Q&A, as indicated by squad2 at the end.

Bert-base refers to the base (neither large nor small) version of Bert. It is also cased, which says it differs between uppercase and lowercase letters.

The Q&A Pipeline

The question-answer pipeline consists of three steps:

1. Initialization of the model and tokenizer
2. Tokenization of queries
3. Pipeline and Prediction

1. **Initialization of the Model and Tokenizer:** As the first step, we will import transformers and use deepset/bert-base-cased-squad2 to initialize our model and tokenizer.

```
from transformers import BertForQuestionAnswering
model = BertForQuestionAnswering.from_pretrained ('deepset/bert-base-

from transformers import AutoTokenizer
tokenizer = AutoTokenizer.from_pretrained ("deepset/bert-base-cased-s
```

2. **Tokenization of Queries:** Now that our tokenizer has been initialized, we can feed it some content to convert into Bert-readable token IDs. Our Bert tokenizer is responsible for converting human-readable content into Bert-friendly data, which we describe to it as token IDs.

To begin, the tokenizer partitions a string into tokens. Bert employs a variety of

special tokens, which are detailed below.

Token	Meaning	Token_ID
[PAD]	The padding token allows us to keep sequences of the same length.	0
[UNK]	Bert uses this phrase when a word is unfamiliar	100
[CLS]	It appears at the beginning of every sequence.	101
[SEP]	Indicates a separator - a point between context and question that appears at the end of sequences.	102

After performing this initial tokenization, we transform this string (token) list into an integer list (token IDs). This is accomplished through the use of an internal dictionary that includes every token understood by Bert.

Each of these tokens corresponds to a distinct integer value; a few of these mappings are shown in the special tokens table above.

3. **Pipeline and Prediction:** Now that we've initialized our model and tokenizer and grasped how our tokenizer converts human-readable strings into lists of token IDs, we can move on to combining them and asking questions!

```
from transformers import pipeline
nlp = pipeline('question-answering', model=model, tokenizer=tokenizer)
```

We can start asking questions:

```
context = ("One of my favorite vacation place is Mexico."
"I really like the weather there because it never gets cold."
"The people are very nice too. They never laugh at my bad Spanish."
"The food is really good. Mexico City is a very interesting place to visit."
"It has some great museums and lots of fascinating old buildings."
"The hotels are too expensive to stay but there are more affordable ones."
"For example, you can stay at one of the beach resorts like Acapulco."
"If you are planning to visit Mexico, you should definitely see the Mayan ruins.")

questions=[
    "what is really good",
    "what is too expensive",
    "which temple you should definitely see"
]

nlp({
    'question': questions[0],
    'context': context
})
```

```
Output : {'score': 0.3535134196281433, 'start': 168, 'end': 192, 'answer':
'The food is really good.'}
```

First, we'll pass along the question and context. While trying to feed this data into a Q&A model, the following format is expected:

```
[CLS] <context> [SEP] <question> [SEP] [PAD] [PAD]
```

The following token ID format will be generated from the input:

```
tokenizer.encode(questions[0], truncation=True, padding = True)
```

```
[101,
 1184,
 1110,
 1103,
 9122,
 12020,
 1282,
 2246,
 11220,
 1110,
 1315,
 5865,
 2246,
 11239,
 1324,
 3550,
 1128,
 1431,
 5397,
 1267,
 102]
```

This is the information that will be passed on to Bert. At this point, Bert uses some linguistic wizardry to detect the answer within the context provided.

Once Bert has made a decision, it returns the span of the answer, that is, the beginning and ending token indexes.

BERT Input Format:

We load both the question and the reference text into the input when we feed a Q&A task into BERT.

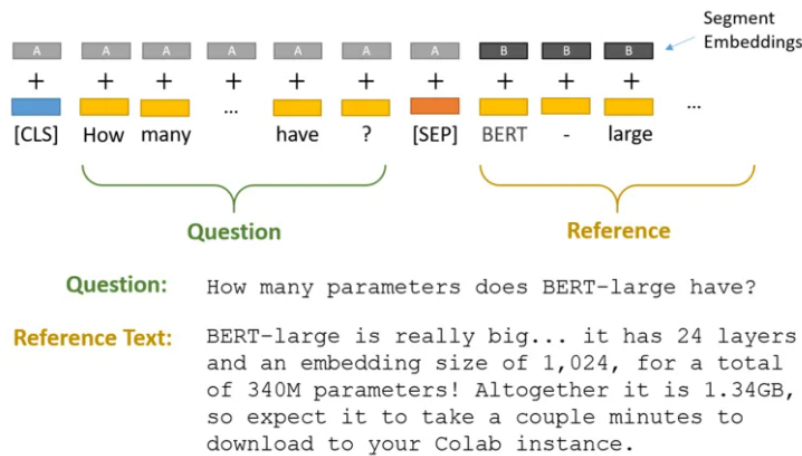


Image By: Chris McCormick

BERT also implements "Segment Embeddings" to distinguish between the question and the reference text. These would be two separate embeddings (for segments "A" and "B") learned by BERT and added to the token hidden states before passing them to the input layer.

Token Classifiers at the Beginning and End

BERT must highlight a "span" of input text and the answer. This is defined by simply predicting which token marks the beginning and which token marks the end of the

answer.

We feed the completed encoding of each token in the text into the beginning token classifier. The start token classifier has a single set of weights that it applies to every word (represented by the blue "start" rectangle in the following illustration).

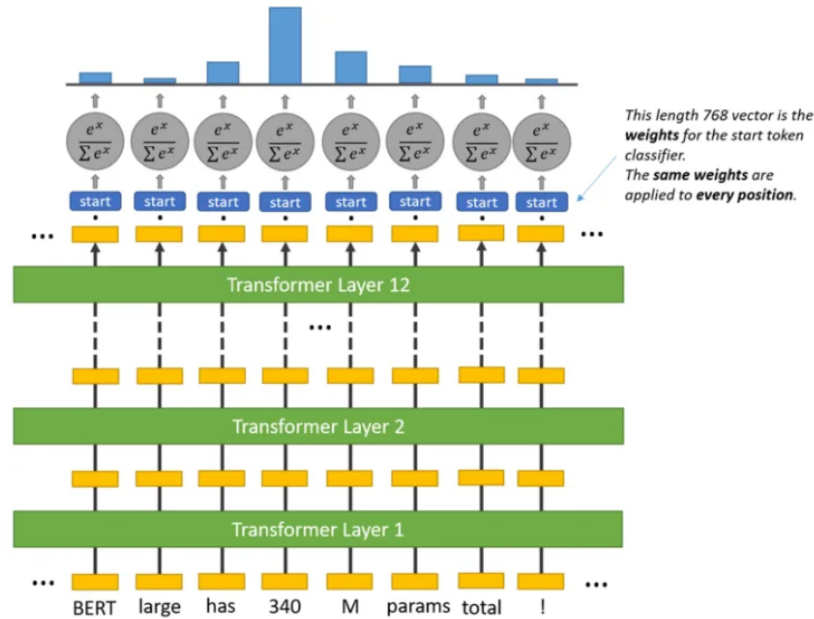


Image By: Chris McCormick

We use the softmax activation to generate a probability distribution for all of the words after taking the dot product of the output embeddings and the "start" weights. We choose the word with the highest probability of being the starting token.

We repeat this procedure for the final token, which has its own weight vector.

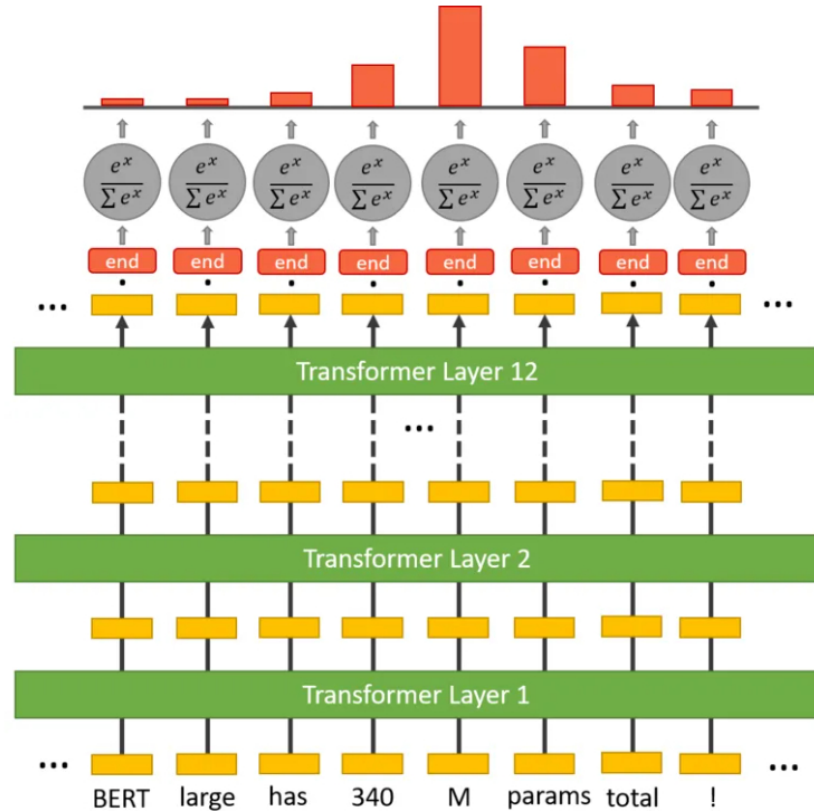








Image By: Chris McCormick

Following that, our pipeline will extract the appropriate token IDs based on Bert's predicted start-end token index. These token IDs will then be decrypted into human-readable text by our tokenizer.

We have accomplished our first Q&A transformer model!

I hope you enjoyed reading this blog! Please click the "like" button and add a comment if you have any thoughts or suggestions for improving this blog.

 Comments



Share your thoughts...

Reply

