Question Answering System

1 ABSTRACT

This project attempts to build Solution that read pdf file and answers the questions based on text paragraphs inside the pdf related to different topics.

2 INTRODUCTION

Reading paragraphs from a pdf file is the ability to read text, process it, and understand its meaning and answer the questions based on the text. This requires an understanding of natural language and semantic/syntactic knowledge. There has recently been a strong increase in the research of question answering, which identifies and extracts answers from text paragraphs.

A goal is to provide a system that can answer questions posed by human's natural language for a given text. This includes reading the paragraph from pdf, which may be a collection of sentences forming a story, or a set of facts forming a knowledge base. The system is expected to read the question and output an answer which may be a single word or a natural language sentence. We are focusing on predicting the right sentence in the paragraph which has the correct answer.

We are building Mobile Application Software for this project all you have to do is upload a pdf file, select methods then ask a question and click on the answer menu after processing you will get the predicted answer.

3 RELATED WORKS

SQUAD: 100,000+ Questions for Machine Comprehension of Text.

This paper analyses the dataset to understand the types of reasoning required to answer the questions, leaning heavily on dependency and constituency trees. It implements a logistic regression model with lexicalized and dependency tree path features. We are trying to implement some of the experiments in this paper for our baseline model.

Attention Is All You Need explained by Jay Alammar in his blog The Illustrated Transformer.

In this paper the authors introduced a new novel neural network called the Transformers which is an attention-based encoder-decoder type architecture. the encoder maps an input sequence into an abstract continuous representation that holds all the learned information of that input. The decoder then takes that continuous representation and step by step generates a single output while also being fed the previous output.

4 PROBLEM

4.1 Problem Statement

User upload pdf file and ask question after processing our system return answer.

4.2 Problem Definition

We are trying to make full production scale solution (Android Application). We are building Algorithm that return answer on uploading pdf file and asking question, based on analysing pdf file our NLP model return answer.

5 PROBLEM FORMULATION

Naive Approach

Here, we can apply naive approach i.e., converting sentences of document into vector and converting questions into vector and apply brute force method to find the cosine similarity between the sentence vector and question vector and find the min angle between them it means find the max cosine value between them and sentence with max cosine value is the answer of the given question.

Steps:

- Scan whole document and map the words as key and value as index (make dictionary).
- Split document into sentences (data frames) and then store each row into list.
- Make vector of this list.

Ex: Document: Age (index: 1), means (2), age (3), at (4), last (5), birthday (6), Accident (7), sudden (8), unforeseen (9), and (10), involuntary (11), event (12), caused (13), by (14), external (15), visible (16), and (17), violent (18)

Suppose sentence is "Age means?"

Sentence Vector: 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Similar make User Question vector (after removing stopping words like what, how, why etc.)

Iterate whole sentence vector and find the cosine similarity with user question vector and find the max value (if the value is high, it means angle between them is minimum and properties between both vectors matches with max probability).

Word Embedding Approach

- Word2Vec Embedding Technique

  In this approach we take input of large corpus words and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space.

  Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space. Word2Vec is a particularly computationally-efficient predictive model for learning word embeddings from raw text.

- Glove Embedding Technique

  Glove is an alternate approach to build word embeddings using matrix factorization techniques on the word-word co-occurrence matrix.

  A large matrix of co-occurrence information is constructed and you count each “word” (the rows), and how frequently we see this word in some “context” (the columns) in a large corpus. Usually, we scan our corpus in the following manner: for each term, we look for context terms within some area defined by a window-size before the term and a window-size after the term. Also, we give less weight for more distant words.

The number of “contexts” is, of course, large, since it is essentially combinatorial in size. So, then we factorize this matrix to yield a lower-dimensional matrix, where each row now yields a vector representation for each word. In general, this is done by minimizing a “reconstruction loss”. This loss tries to find the lower-dimensional representations which can explain most of the variance in the high-dimensional data.

Bert

Instead of looking at words in isolation, BERT, a transformer-based model attempts to use the context of words to get embeddings. BERT uses several concepts of deep learning come up with a model that looks at context in a bidirectional fashion, leveraging information from the entire sentences as a whole through self-attention.

Example: Suppose Question is "How I can cross a road when there is no zebra crossing." and this question very similar to query "How I can cross road on zebra crossing." and there is a chance that above two method give wrong answers but could handle that question. And we don't remove stopping words in the document because if it removes word like in above example "no" meaning of whole sentence become change. Bert uses stopping words to make relations between words.

6 METHODOLOGY

In the Frontend (Android App) we are making 3 sections (activity pages) i.e., upload pdf, method selection and last ask a question. In the upload section user upload a pdf file that uploaded to firebase then the user moves to the next activity page on selecting methods a method code is stored in a public static variable that is accessible in any java class then the user moves to the question section and type a question and after clicking answer button by use of okhttp3 we make a POST request and send pdf file name, method code and question to the flask server and after processing flask server return result(answer) and we print that answer in the text view.

Also, set up firebase storage for storing pdf and add URL parameter in firebase remote config because we don't need to update ngrok public URL in the frontend code just update URL in remote config in firebase and fetch that remote config URL from the android side.

In the backend server, we are making it with the help of flask and ngrok that provide public URL for a very limited time and copy that URL and paste it to firebase remote config URL parameter and publish.

In the flask server, we have method code, pdf and question, first of all, convert pdf to text using pdfplumber library and according to function code executes suitable method function and pass text and question as a parameter and after execution, it returns an answer.

We trying 4 methods Naive method, Word Embedding Word2Vec and Glove, Bert.

In the Naive Approach we convert sentences into vector and converting questions into vector and apply brute force method to find the cosine similarity between the sentence vector and question vector and find the min angle between them it means find the max cosine value between them and sentence with max cosine value is the answer of the given question.

In the Word Embedding Technique Word2vec we take input of large corpus words and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space. Word2Vec is a particularly computationally-efficient predictive model for learning word embeddings from raw text. Actually, Word2Vec is a simple neural network with a single hidden layer, and like all neural networks, it has weights, and during training, its goal is to adjust those weights to reduce a loss function. However, Word2Vec is not going to be used for the task it was trained on, instead, we will just take its hidden weights, use them as our word embeddings, and toss the rest of the model (this came out a little bit evil).

In the Word Embedding Technique Glove, it can build word embeddings using matrix factorization techniques on the word-word co-occurrence matrix. A large matrix of co-occurrence information is constructed and you count each “word” (the rows), and how frequently we see this word in some “context” (the columns) in a large corpus. Usually, we scan our corpus in the following manner: for each term, we look for context terms within some area defined by a window-size before the term and a window-size after the term. Also, we give less weight for more distant words. The number of “contexts” is, of course, large, since it is essentially combinatorial in size. So, then we factorize this matrix to yield a lower-dimensional matrix, where each row now yields a vector representation for each word. In general, this is done by minimizing a “reconstruction loss”. This loss tries to find the lower-dimensional representations which can explain most of the variance in the high-dimensional data.

In Bert approach Instead of looking at words in isolation, BERT, a transformer-based model attempts to use the context of words to get embeddings. BERT uses several concepts of deep learning come up with a model that looks at context in a bidirectional fashion, leveraging information from the entire sentences as a whole through self-attention. Example: Suppose Question is "How I can cross a road when there is no zebra crossing." and this question very similar to query "How I can cross road on zebra crossing." and there is a chance that above two method give wrong answers but could handle that question. And we don't remove stopping words in the document because if it removes word like in above example "no" meaning of whole sentence become change. Bert uses stopping words to make relations between words.

Steps included in the Bert approach are:

- Installing transformers
- Load Fine-Tuned BERT-large
- Load the tokenizer
- Apply the tokenizer to the pdf text, treating them as a text-pair.

- We can concatenate the question and pdf text together, but BERT still needs a way to distinguish them. BERT has two special "Segment" embeddings, one for segment "A" and one for segment "B". Before the word embeddings go into the BERT layers, the segment A embedding needs to be added to the question tokens, and the segment B embedding needs to be added to each of the pdf text tokens. These handled by the transformer library and all we need to do is specify '0' and '1' for the token.
- Feed pdf text and question into model
- Then, we find the highest 'start' and 'end' scores also find sits position and the combine the tokens in the answer and return answer.

7 EXPERIMENTS

Bert is a really powerful model for tackling a question-answering problem. However, it comes up with the limitation of 512 tokens and the documents were really longer than 512 tokens. In order to handle this limitation, I wrote the function "expand_split_sentences", which split and expand sentences i.e., it makes paragraphs with lesser than 512 tokens and makes data frames of that paragraph. In this, more than one data frame contains the correct answer so we will find the best answer by finding the max start score.

In word2vec we are using “w2vecmodel” model, for glove approach we are using “glovemodel” model. For Bert we are using “bert-large-uncased-whole-word-masking-finetuned-squad” model that trained on squad dataset, 24 transformer layers with 336M parameter and 30,000 vocabulary size.

Accuracy of BERT on SQuAD v1.1 is 93.5 % for one set of QA with reference text having at most 512 tokens.

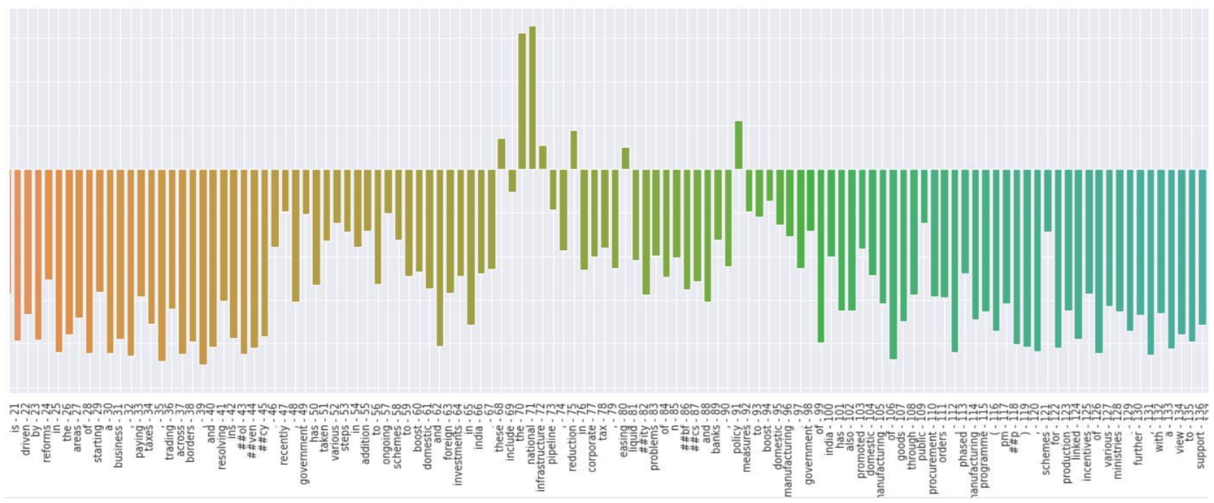
On Testing Our Model:

Query: What were the various steps taken by the government to boost domestic and foreign investment in India?

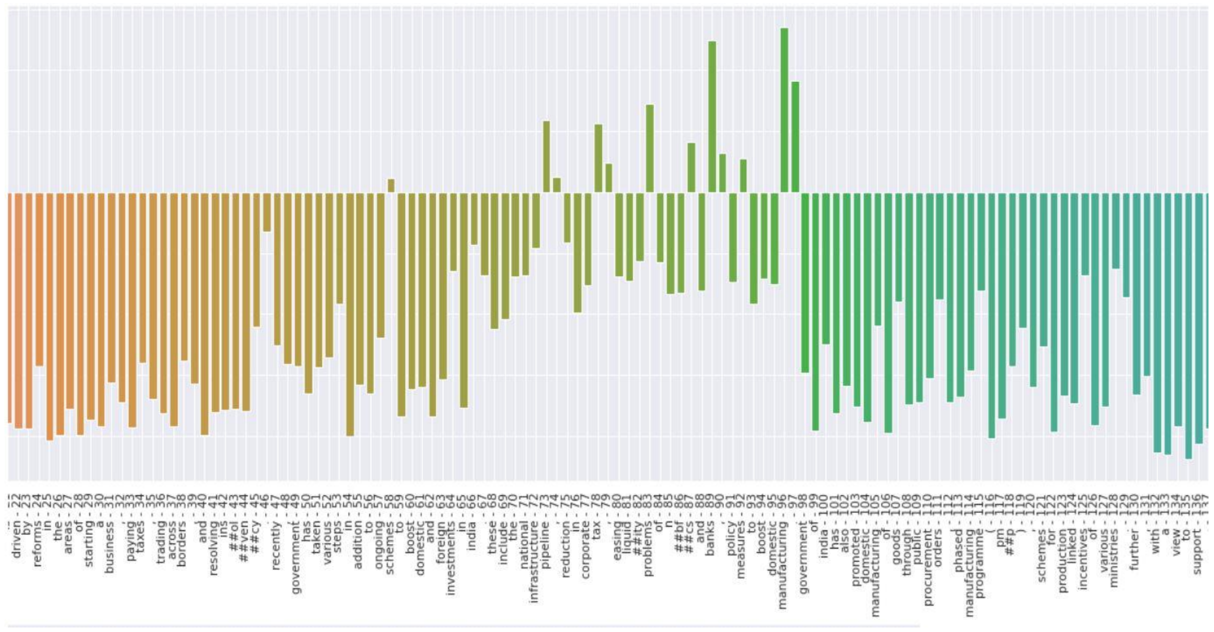
Predicted Answer: national infrastructure pipeline, reduction in corporate tax, easing liquidity problems of nbfcs and banks, policy measures to boost domestic manufacturing

8 RESULTS AND DISCUSSION

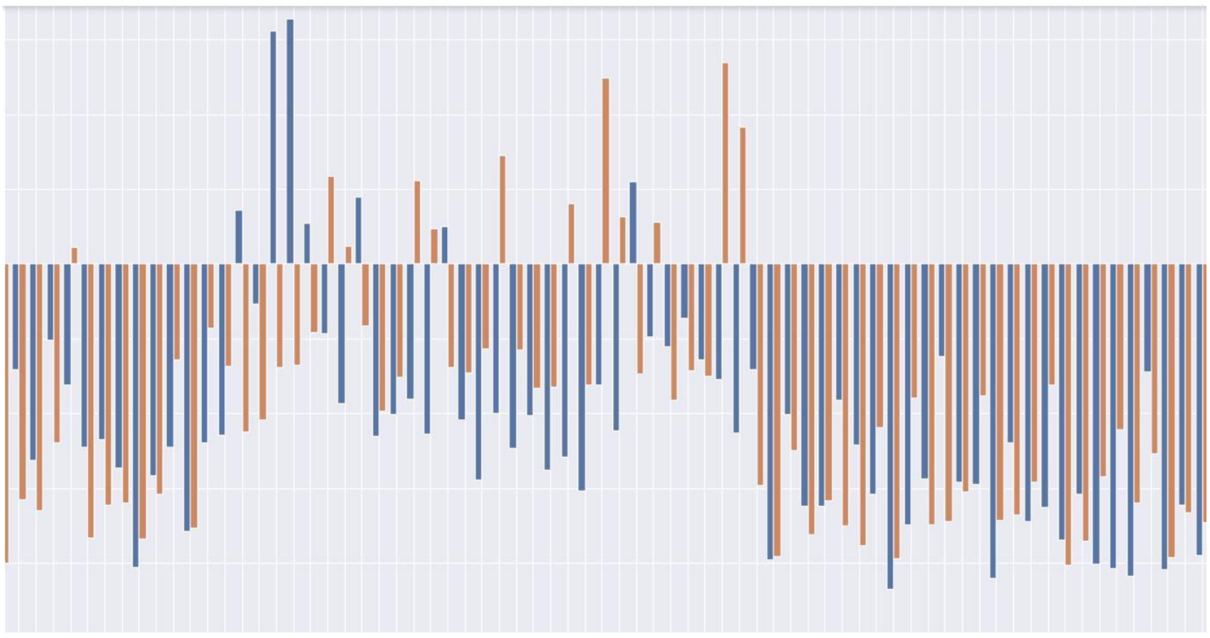
Our Model Visualisation for Start Score:



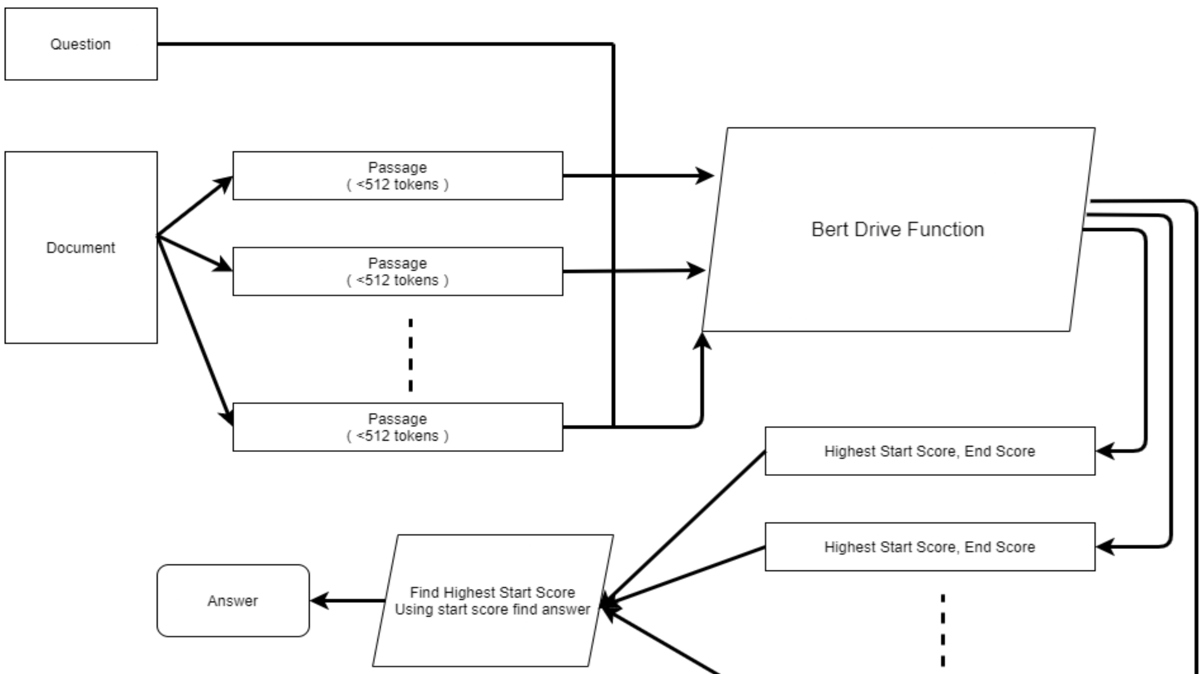
Our Model Visualisation for End Score:



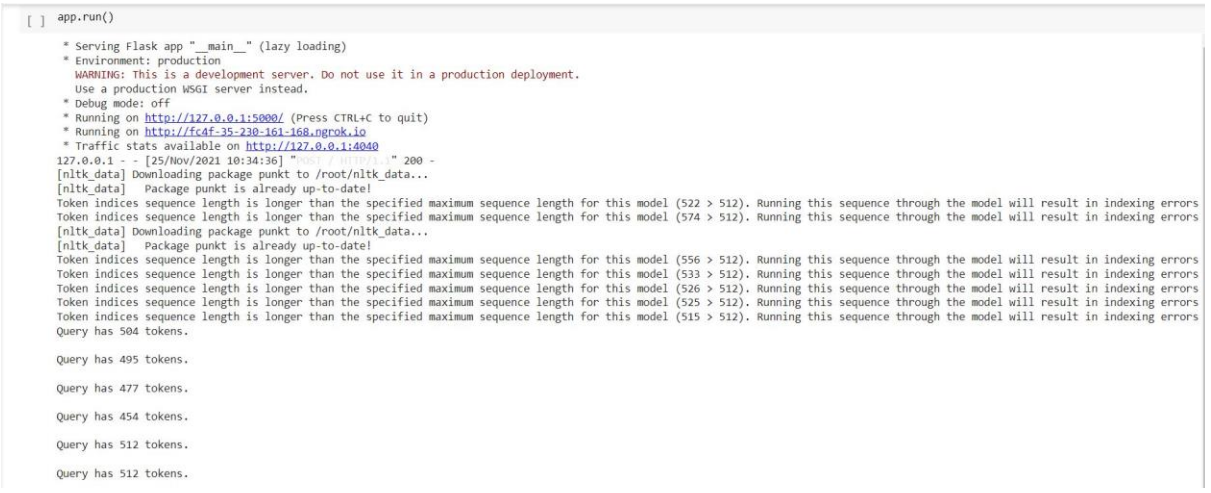
Our Model Visualisation for Start and End Score:



Data Flow Diagram

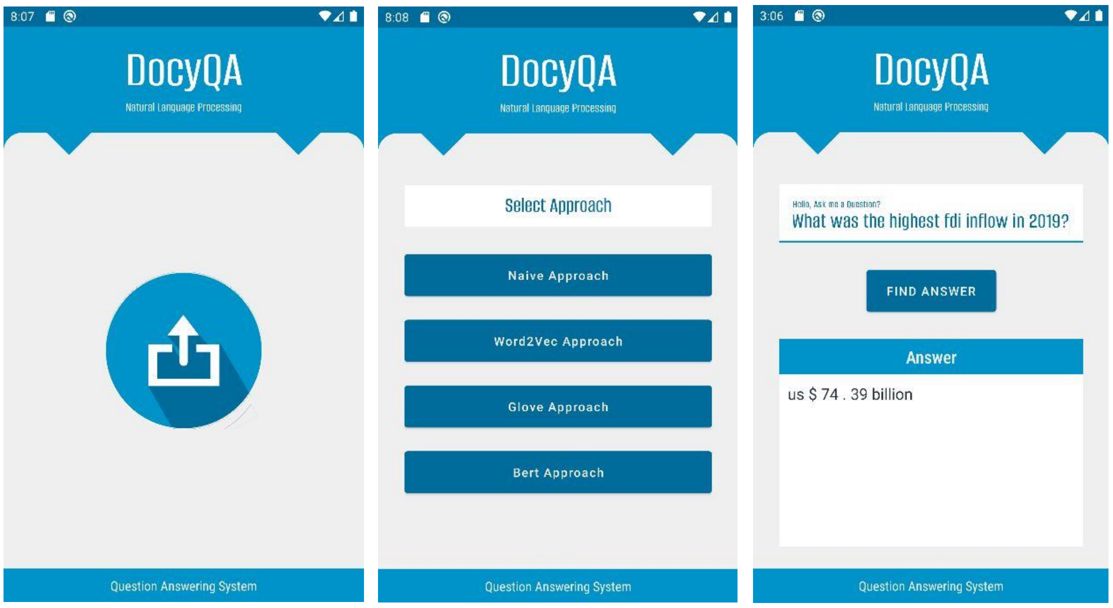

Question
Document
Passage (<512 tokens)
Passage (<512 tokens)
Passage (<512 tokens)
Bert Drive Function
Highest Start Score, End Score
Highest Start Score, End Score
Highest Start Score, End Score
Find Highest Start Score Using start score find answer
Answer

Here is the Screenshot of Flask Server



```
[ ]  app.run()

      * Serving Flask app "__main__" (lazy loading)
      * Environment: production
        WARNING: This is a development server. Do not use it in a production deployment.
        Use a production WSGI server instead.
      * Debug mode: off
      * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
      * Running on http://fc4f-35-230-161-168.ngrok.io
      * Traffic stats available on http://127.0.0.1:4040
     127.0.0.1 - - [25/Nov/2021 10:34:36] "POST / HTTP/1.1" 200 -
     [nltk_data] Downloading package punkt to /root/nltk_data...
     [nltk_data]   Package punkt is already up-to-date!
     Token indices sequence length is longer than the specified maximum sequence length for this model (522 > 512). Running this sequence through the model will result in indexing errors
     Token indices sequence length is longer than the specified maximum sequence length for this model (574 > 512). Running this sequence through the model will result in indexing errors
     [nltk_data] Downloading package punkt to /root/nltk_data...
     [nltk_data]   Package punkt is already up-to-date!
     Token indices sequence length is longer than the specified maximum sequence length for this model (556 > 512). Running this sequence through the model will result in indexing errors
     Token indices sequence length is longer than the specified maximum sequence length for this model (533 > 512). Running this sequence through the model will result in indexing errors
     Token indices sequence length is longer than the specified maximum sequence length for this model (526 > 512). Running this sequence through the model will result in indexing errors
     Token indices sequence length is longer than the specified maximum sequence length for this model (525 > 512). Running this sequence through the model will result in indexing errors
     Token indices sequence length is longer than the specified maximum sequence length for this model (515 > 512). Running this sequence through the model will result in indexing errors
     Query has 504 tokens.

     Query has 495 tokens.

     Query has 477 tokens.

     Query has 454 tokens.

     Query has 512 tokens.

     Query has 512 tokens.
```

Here is the Screenshot of UI


8:07
DocyQA
Natural Language Processing
Question Answering System
8:08
DocyQA
Natural Language Processing
Select Approach
Naive Approach
Word2Vec Approach
Glove Approach
Bert Approach
Question Answering System
3:06
DocyQA
Natural Language Processing
Hello, Ask me a Question?
FIND ANSWER
Answer
Question Answering System

What was the highest fdi inflow in 2019?

us $ 74 . 39 billion

9 CONCLUSIONS

We observed the Bert approach performed qualitatively better compared to all other approaches but it could take more processing time as compared to other approaches. The Bert model is limited to 512 tokens only then, we were made a new function called expand_split_sentences to process it for whole text in the pdf file. The Simple Naive take minimum time in processing but the result is bad. So, Our Android Application is performed well when we choose Bert option in the method selection activity page but worst in the naive approach.

10 REFERENCES

- https://arxiv.org/pdf/1810.04805.pdf

- https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
- https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html
- http://jalammar.github.io/illustrated-transformer/
- http://jalammar.github.io/illustrated-bert/
- https://www.youtube.com/playlist?list=PLam9sigHPGwOBuH4_4fr-XvDbe5uneaf6

11 PROJECT LINKS

- https://colab.research.google.com/drive/1TobzyLKmTooW3a-L1kdomeb8gX_5gDcb?usp=sharing
- https://github.com/SatyamSoni23/DocyQA