**PROJECT REQUIREMENT AND SPECIFICATION**

**ON**

**QUESTION ANSWERING SYSTEM**

**CSE  VTH SEMESTER MINI PROJECT**

**2021-2022**



**Submitted to:       Submitted by:**

DR. GURUPRASAD/ MR KUSHAGRA TRIVEDI

MR. PANKAJ KUMAR      University Roll. No.: 2015079

CSE-DS- Vth Sem

GUIDED BY:

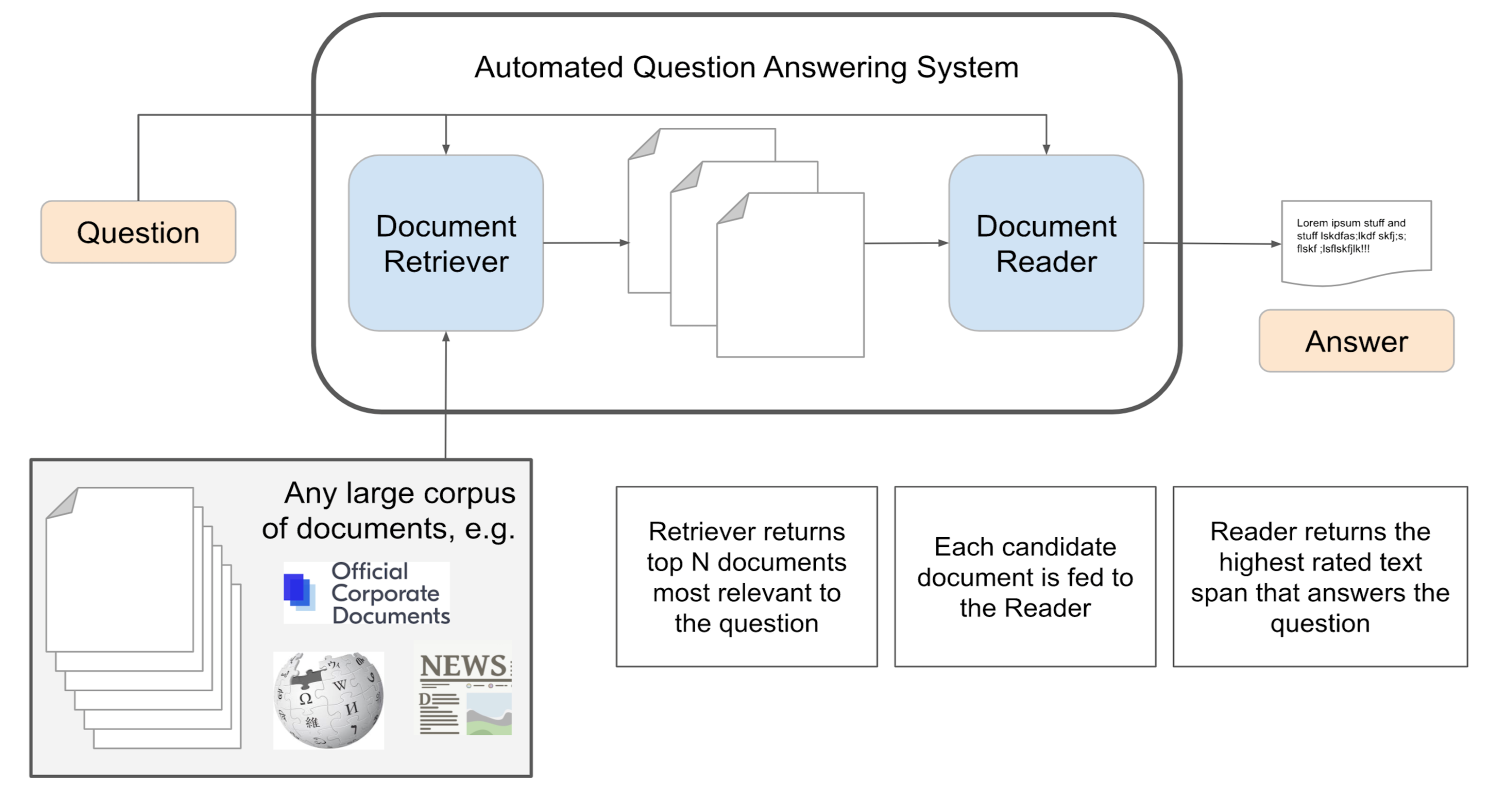
MR. ASHWINI KUMAR

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GRAPHIC ERA DEEMED UNVERSITY, DEHRADUN**

* 1. **About Project**

Asking a question to a machine and receiving an answer was always the stuff of sci-fi in the not too distant past. Now, things have changed, and we find ourselves using Q&A systems everywhere without even realizing it. Google search is the best example although in most cases Google is used to find information and will simply point you in the right direction, many other queries are direct questions to which Google often provides direct answers.We are building an IR-based QA system.Covering the general design of these systems, which typically require two main components: the document retriever (a search engine) that selects the n most relevant documents from a large collection, and a document reader that processes these candidate documents in search of an explicit answer span.



**By the end of this post we'll have a working IR-based QA system, with BERT**

**as the document reader and Wikipedia's search engine as the document**

**retriever**

**Hugging Face’s Transformers:**

First things first, modern NLP is dominated by these incredible models called **transformers**. These models are brilliant, and a comparatively recent development (the first paper describing a transformer appeared in 2017).

Second, the real-world implementation of transformers is carried out almost exclusively using a library called transformers built by an incredible collection of people that refer to themselves as HuggingFace.

There are many reasons that the transformers library is so popular. But there are three factors, in particular, that come to mind:

1. **Open-source** — the library is open-source, and it has an incredibly active community
2. **Ease of use** — we can begin using state-of-the-art models from Google, OpenAI, and Facebook in around five lines of code
3. **Models**— the library has an unimaginably huge number of models available that we can simply download and begin working with

**1.2 METHODOLOGY:**

First of all we are going to install transformers and this library relies on pytorch

and tensorflow running in the background.

Now we are going to set our Q-A transformer for which we are going to select a

Model from hugging face multiple pre-trained models.We are going to use

**deepset/bert-base-cased-squad2** model for our project.

The model has been built by Deepset.AI. and has been pre-trained for Q&A on the

SQuAD 2.0 dataset.

Now we start our question-answering process which at its core consists of three steps:

1. Model and tokenizer initialization
2. Query tokenization
3. Pipeline and Prediction

**Model and Tokenizer Initialization** — this is our very first step, here we will import transformers and initialize our model and tokenizer using deepset/bert-base-cased-squad2.

**Input Data Tokenization** — We’ve initialized our tokenizer, and now it’s time to feed it some text to convert into Bert-readable token IDs.Our Bert tokenizer is in-charge of converting human-readable text into Bert-friendly data, called token IDs.First, the tokenizer takes a string and splits it into tokens.

Different tokens used with their meanings are as follows:

| **Token** | **Meaning** | **Token ID** |
| --- | --- | --- |
| **[PAD]** | Padding token, allows us to maintain same-length sequences (512 tokens for Bert) even when different sized sentences are fed in | 0 |
| **[UNK]** | Used when a word is unknown to Bert | 100 |
| **[CLS]** | Appears at the start of every sequence | 101 |
| **[SEP]** | Indicates a seperator - used to indicate point between context-question and appears at end of sequences | 102 |
| **[MASK]** | Used when masking tokens, for example in training with masked language modelling (MLM) | 103 |

Every one of these tokens maps to a unique integer value, a few of these mappings

are listed in the special tokens table above. It is these token IDs that are fed into

Bert.The tokenizer converts our input text into 2541 tokens, but this far exceeds the

maximum number of tokens that can be fed to the model at one time. Most BERT-

esque models can only accept 512 tokens at once, thus the (somewhat confusing)

warning above (how is 10 > 512?). This means we'll have to split our input into

chunks and each chunk must not exceed 512 tokens in total.When working with

Question Answering, it's crucial that each chunk follows this format:

[CLS] question tokens [SEP] context tokens [SEP]

This means that, for each segment of a Wikipedia article, we must prepend the

original question, followed by the next "chunk" of article tokens.Each of these

chunks can now be fed to the model without causing indexing errors. We'll get an

"answer" for each chunk; however, not all answers are useful, since not every

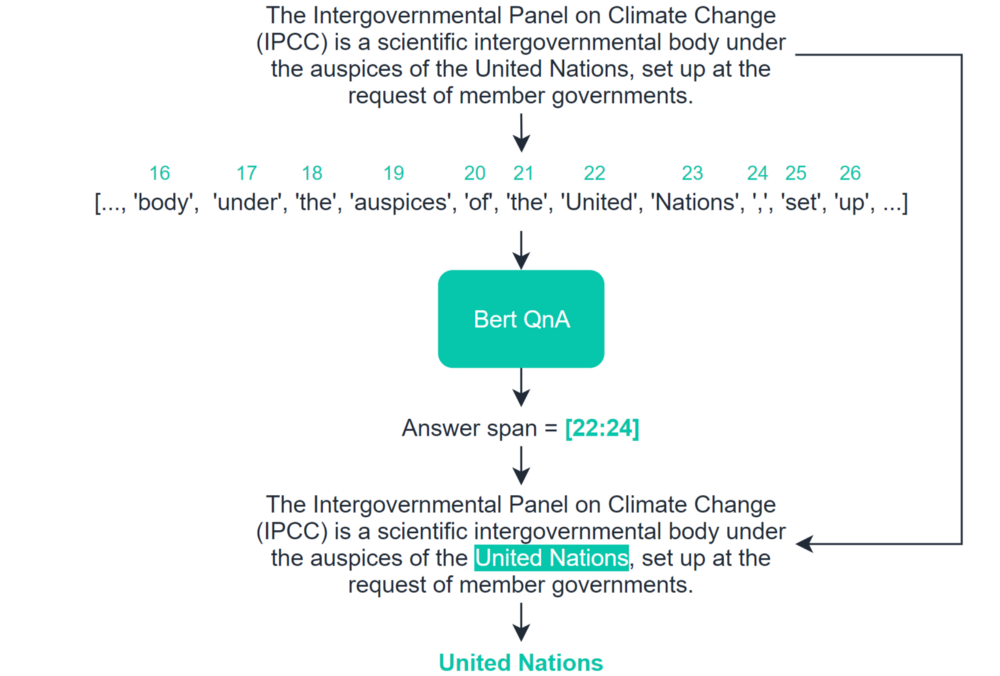
segment of a Wikipedia article is informative for our question. The model will

return the [CLS] token when it determines that the context does not contain an

answer to the question.

**Pipeline and Prediction** — Now we’ve initialized our model and tokenizer, and understood how our tokenizer is converting our human-readable strings into lists of token IDs, we can move to integrating this and asking questions!

The Pipeline Function can be understood using this below diagram:

****

That’s it, we’ve built our first Q&A transformer model! . Most likely, there are one more thing we will need to consider, such as:

* **Fine-tuning our model**— the pre-trained models are great but can struggle when applied to Q&A where the language/formatting is slightly different (like financial terminology, or tweets).I would like to mention that I tried fine tuning the model on google collab but the GPU usage limit exceeded for three times I tried training it putting more than 20 hours of training.

These are some additional steps that will likely prove crucial when moving ahead and developing real-world use-cases — however, they’re not always required — and what we have built here represents the very core of the Q&A process.

**There we have it! A working QA system on Wikipedia articles.**

**Requirements of Project:**

* 1. **Hardware Requirement:**
* Processor: min 1 GHz ,recommended 1.60 GHz or more
* Ethernet connection or a wi-fi
* Hard Drive: min 32GB ,recommended 64 GB or more
* Memory(RAM):8 GB recommended
* OS:WINDOWS 10
* System type: 64 bit recommended.
* GPU:16GBrecommended

1. **Software Requirement:**

* Python 3s version
* Google Collab Notebook
* Microsoft Excel

**3. Modules of Project**

* Hugging Face transformer Installation(deepset/bert-base-cased-squad2 model )Extraction
* Model and tokenizer initialization
* Query tokenization
* Pipeline and Prediction

**4.BRIEF MOTIVATION:**

I want to thank Computer Science Engineering Department who provided me this

opportunity to do the mini project .I also want to thank my teachers who provided

the lecture to understand all important topics related to project and taught us to

write and run the program, they also took doubt sessions to clear our doubts.

Completing this project has  build a confidence in me and now i am able to

perform more projects like these.

**5.  REFERENCES:**

**1.**Github.com for SquaD 2.0 datasets download

**2.**Hugging Face Documentation

**3.**Google.com for some explanations

**4.**Towards DataScience Documentation

**THANK YOU!**