

**University of New Haven**  
**Tagliatela college of Engineering**  
**Masters in data science**



**Natural Language Processing**

DSCI-6004-01

Date: 12/12/2022

By,

**Riz Amatya**

**Abhinaya Movva**

**Harsha Kolla**

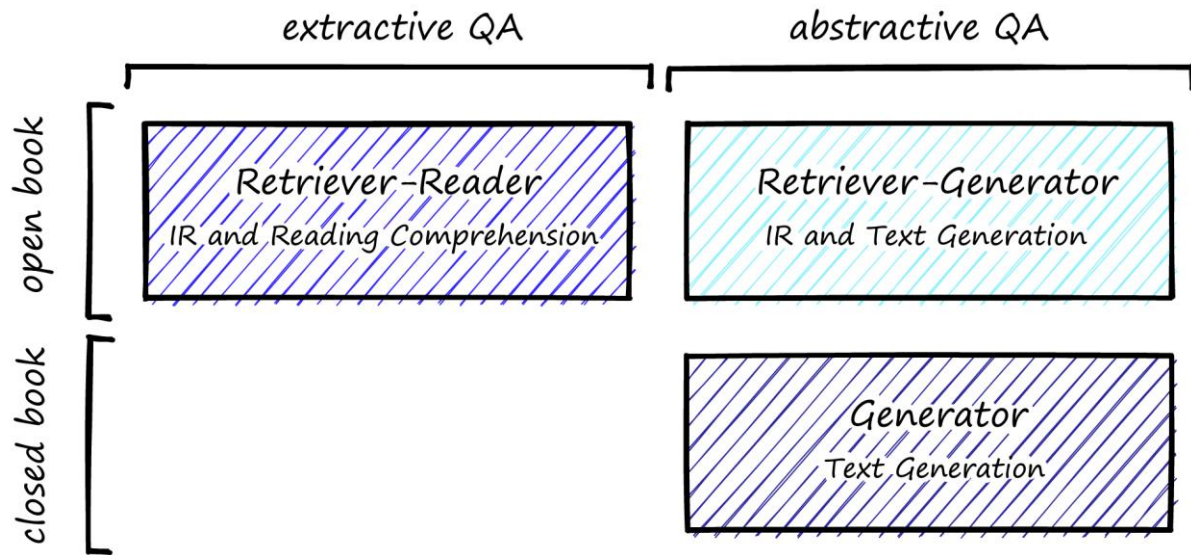
**Supervisor : Vahid Behzadan**

## **Motivation:**

Natural language processing tasks such as machine reading comprehension and question-answering are crucial. Due to their outstanding performance in a variety of NLP tasks, Pre-trained Contextual Embeddings (PCE) models like Embeddings from Language Models(ELMo) and Bidirectional Encoder Representations from Transformers (BERT) have recently received a lot of attention. In this study, we adopted the various model and worked to improve its performance on the Stanford Question Answering Dataset (SQuAD 2.0). We carefully evaluated the performance of a number of output designs in comparison to all the models.

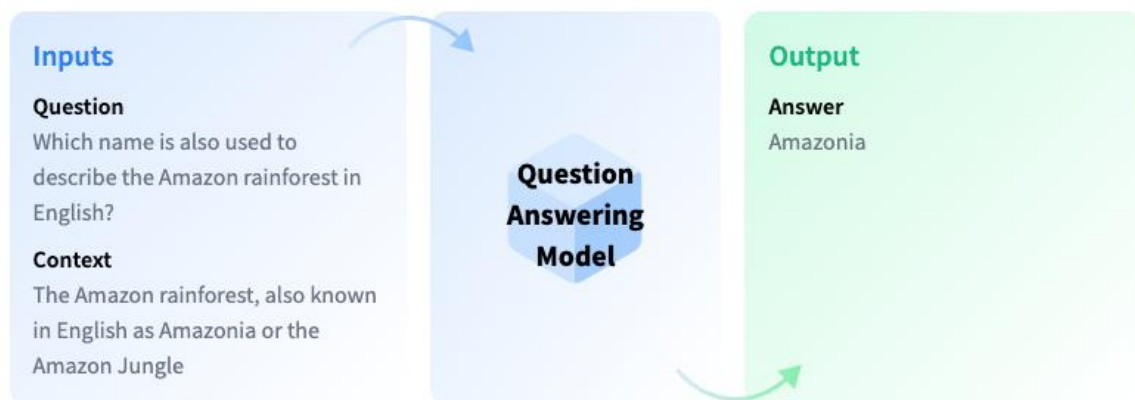
## **Introduction:**

Question Answering models can retrieve the answer to a question from a given text, which is useful for searching for an answer in a document. Some question answering models can generate answers without context. We can use Question Answering (QA) models to automate the response to frequently asked questions by using a knowledge base (documents) as context. As a business use case, answers to customer questions can be drawn from those documents.



There are different QA variants based on the inputs and outputs:

- **Extractive QA:** The model extracts the answer from a context. The context here could be a provided text, a table or even HTML! This is usually solved with BERT-like models
- **Open Generative QA:** The model generates free text directly based on the context
- **Closed Generative QA:** In this case, no context is provided. The answer is completely generated by a model, as shown in the figure below



The schema above illustrates extractive, open book QA. The model takes a context and the question and extracts the answer from the given context. We can also differentiate QA models depending on whether they are open-domain or closed-domain. Open-domain models are not restricted to a specific domain, while closed-domain models are restricted to a specific domain.

## **Objective:**

We formulate the problem as follows → Given a question  $q$  and context  $c$ , identify spans within  $c$  that answer  $q$ . The schema below demonstrates this paradigm:

### **Passage Sentence**

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity.

### **Question**

What causes precipitation to fall?

### **Answer Candidate**

gravity

## **DataSet:**

We use the **Stanford Question Answering Dataset (SQuAD)** for pre-training, fine-tuning, and evaluating our models. **SQuAD** is a reading comprehension dataset, consisting of questions posed by crowd workers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage.



We train pre-trained models based on the transformer architecture for this task, as explained in the next section.

## **Methodology:**

We train the following pre-trained models based on the transformer architecture for this task:

1. **BERT:** Pre-training of Deep Bidirectional Transformers for Language Understanding (base-uncased variant)
2. **BERT:** Pre-training of Deep Bidirectional Transformers for Language Understanding (base-cased variant)
3. **RoBERTa:** A Robustly Optimized BERT Pretraining Approach
4. **ELECTRA:** Pre-training Text Encoders as Discriminators Rather Than Generators
5. **DeBERTa:** Decoding-enhanced BERT with Disentangled Attention

The transformer works as follows:

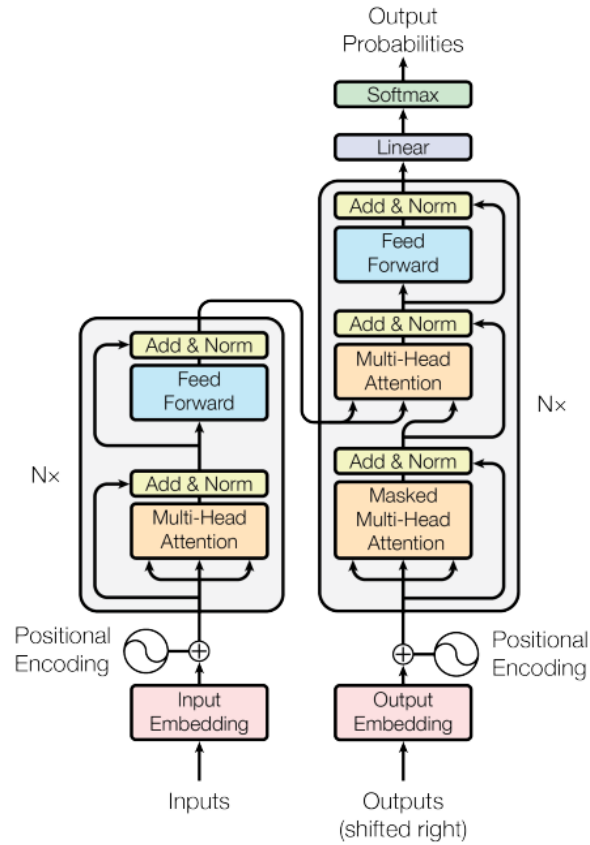
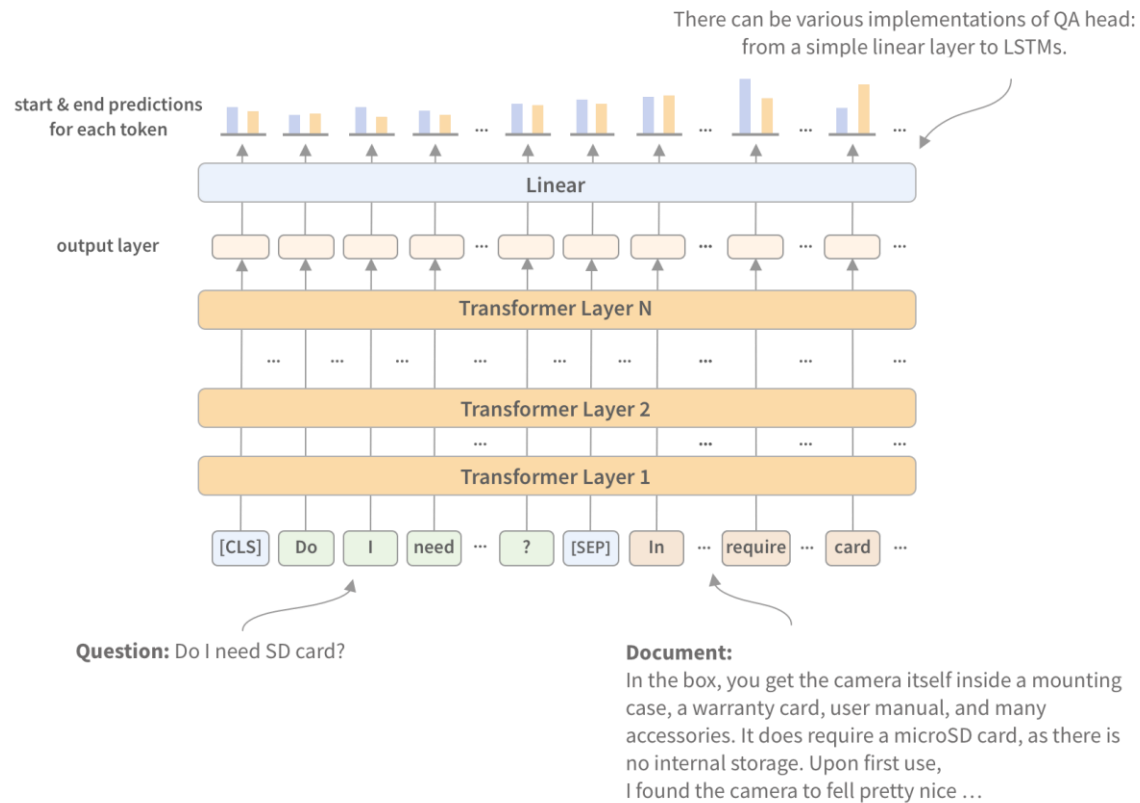


Figure 1: The Transformer - model architecture.

We implement these models in the following framework for extractive span-based question answering:



We implement the above models in Python using the following libraries:

- Huggingface Transformers
- Happy Transformers
- Huggingface Datasets


## Tools:

We have used the following libraries,

- Python
- Numpy
- Pandas
- Torch
- Transformers
- Google colab

## Evaluation Methodology:

- The metric that is used most often used to measure the quality of a model is F1 Score. This metric calculates the average difference between the expected and actual answers.
- We try to compare it with various other models and determine how our model behaves in different scenarios.

 result

```
{'exact': 89.3,  
 'f1': 92.7106662274923,  
 'total': 1000,  
 'HasAns_exact': 89.3,  
 'HasAns_f1': 92.7106662274923,  
 'HasAns_total': 1000,  
 'best_exact': 89.3,  
 'best_exact_thresh': 0.9999721050262451,  
 'best_f1': 92.7106662274923,  
 'best_f1_thresh': 0.9999721050262451,  
 'total_time_in_seconds': 16.367767864000143,  
 'samples_per_second': 61.09568563710119,  
 'latency_in_seconds': 0.016367767864000144}
```



## Results:

We tabulate the results of all experiments as follows:

Model	Version	F1 score	Throughput (average samples processed per second)	Latency (seconds)
BERT	bert-base-uncased-squad2	88.4405	59.59	0.0167
BERT	bert-base-cased-squad2	88.1765	61.49	0.0162
RoBERTa	roberta-base-squad2	92.7439	49.07	0.0203
ELECTRA	electra-base-squad2	91.0828	61.91	0.0161
DeBERTa	deberta-v3-base-squad2	92.6123	39.22	0.0254

We observe the following from the results:

- The RoBERTa model performs the best as measured by the F1 score, with a best F1 score of 92.74%
- However, RoBERTa is significantly slower than the other other models, with a throughput of just 49 samples per second as compared to ~60 samples per second by the other models
- We observe that the DeBERTA model, in spite of performing comparably to RoBERTa, has significantly lower throughput

## **Conclusions:**

In this project, we formulated the extractive question answering problem as: given a question  $q$  and context  $c$ , identify spans within  $c$  that answer  $q$ . We used the Stanford Question Answering Dataset (SQuAD), a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, for pre-training, fine-tuning, and evaluating our models.

Then, we applied the pre-trained models BERT, RoBERTa, ELECTRA, and DeBERTa, all based on the transformer architecture, for this task. We trained these models on a train split of the SQuAD dataset, keeping aside a separate set for validating the model performance on unseen data.

We calculated the F1 score, defined as the harmonic mean of the Precision and the Recall, for each model on the held out validation set, and observed that the RoBERTa model performs the best as measured by the F1 score, but the BERT model has the best latency and highest throughput. Hence, we conclude that the RoBERTa model is best suited for this task.

## **References:**

- [Attention Is All You Need](#)
- [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#)
- [RoBERTa: A Robustly Optimized BERT Pretraining Approach](#)
- [ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators](#)
- [DeBERTa: Decoding-enhanced BERT with Disentangled Attention](#)

## **Github link :**

[https://github.com/harshach729/NLP\\_finalproject](https://github.com/harshach729/NLP_finalproject)