

Introduction to NLP

Term Project -

Question-Answering

Team 16 :-

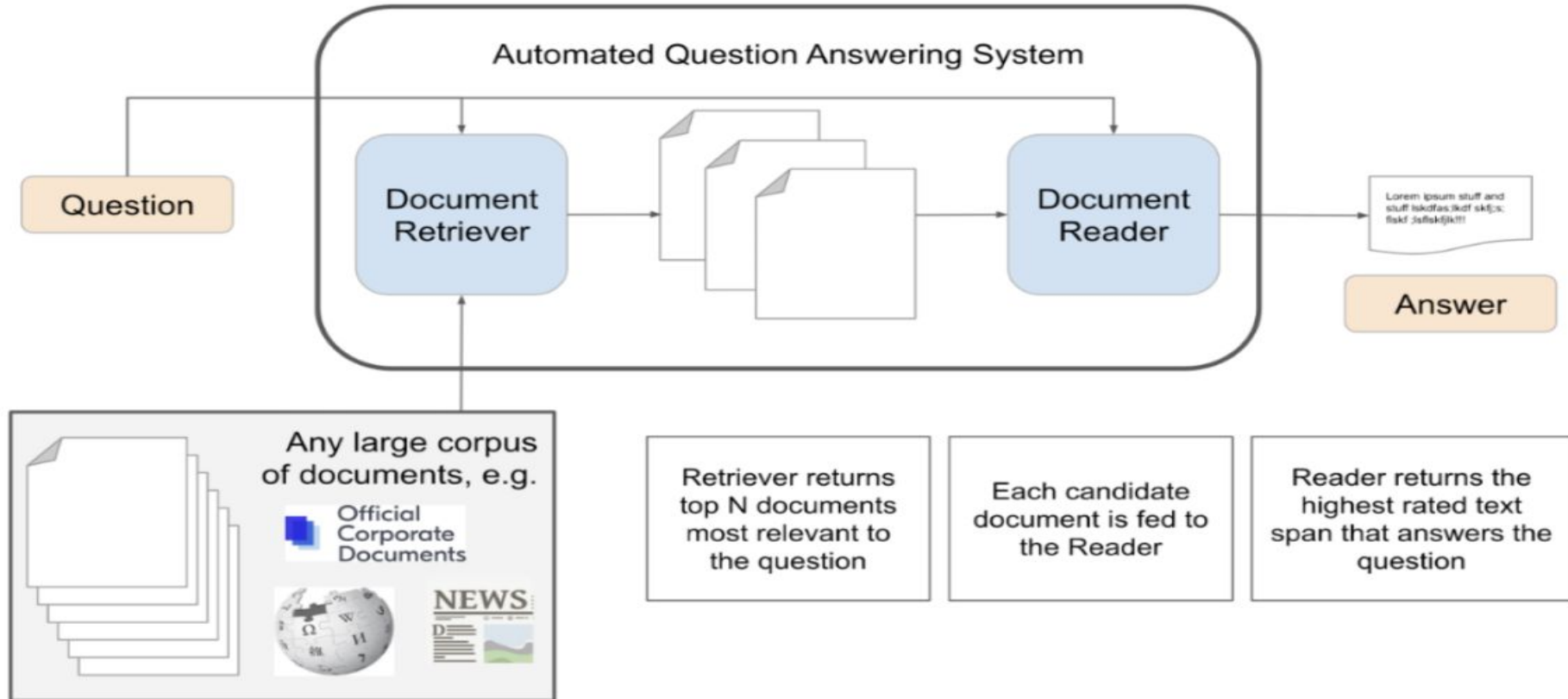
Rahul Padhy (2022201003)

Arun Das (2022201021)

Introduction

- Building up an **Extractive, Open** (extraction of the answer from a context) QnA Model, by investigation of naive approaches (TF-IDF and Word2Vec encodings) and then fine-tuning transformer-based models (BERT, specifically), subsequently followed by testing upon unseen data.
- Datasets used - SQuAD 1.0 and SQuAD 2.0.
- **SQuAD 1.1**, contains 100,000+ question-answer pairs on 500+ articles. **SQuAD2.0** combines the 100,000 questions in **SQuAD1.1** with over 50,000 unanswerable questions written adversarially by crowdworkers to look similar to answerable ones.
- Evaluation metrics used - **Accuracy** for **Document Retrieval** based techniques (using **Word2Vec** and **TF-IDF** encodings) and **F1-Score** and **HuggingFace Pipeline score** for BERT models.

Document Retrieval based Approach



Generic IR QA system

Brief overview of the use of Word2Vec and TF-IDF encodings in Document Retrieval based approaches

- Documents are encoded and questions are transformed into vectors using TF-IDF or Word2Vec.
- Search for a question is performed by comparing with the document vectors.
- The most similar document (NearestNeighbor) is returned.
- Since we have multiple documents and the answer can be in 1 or more documents. So, the retriever should return 'k' documents (where, $k > 1$), as it's not complete / fair to return only 1 document.
- Both Word2Vec and TF-IDF are only able to compute similarity between questions and documents that present the same words, so they are unable to capture synonyms properly.
 - They cannot understand the question's context or the meaning of the words.

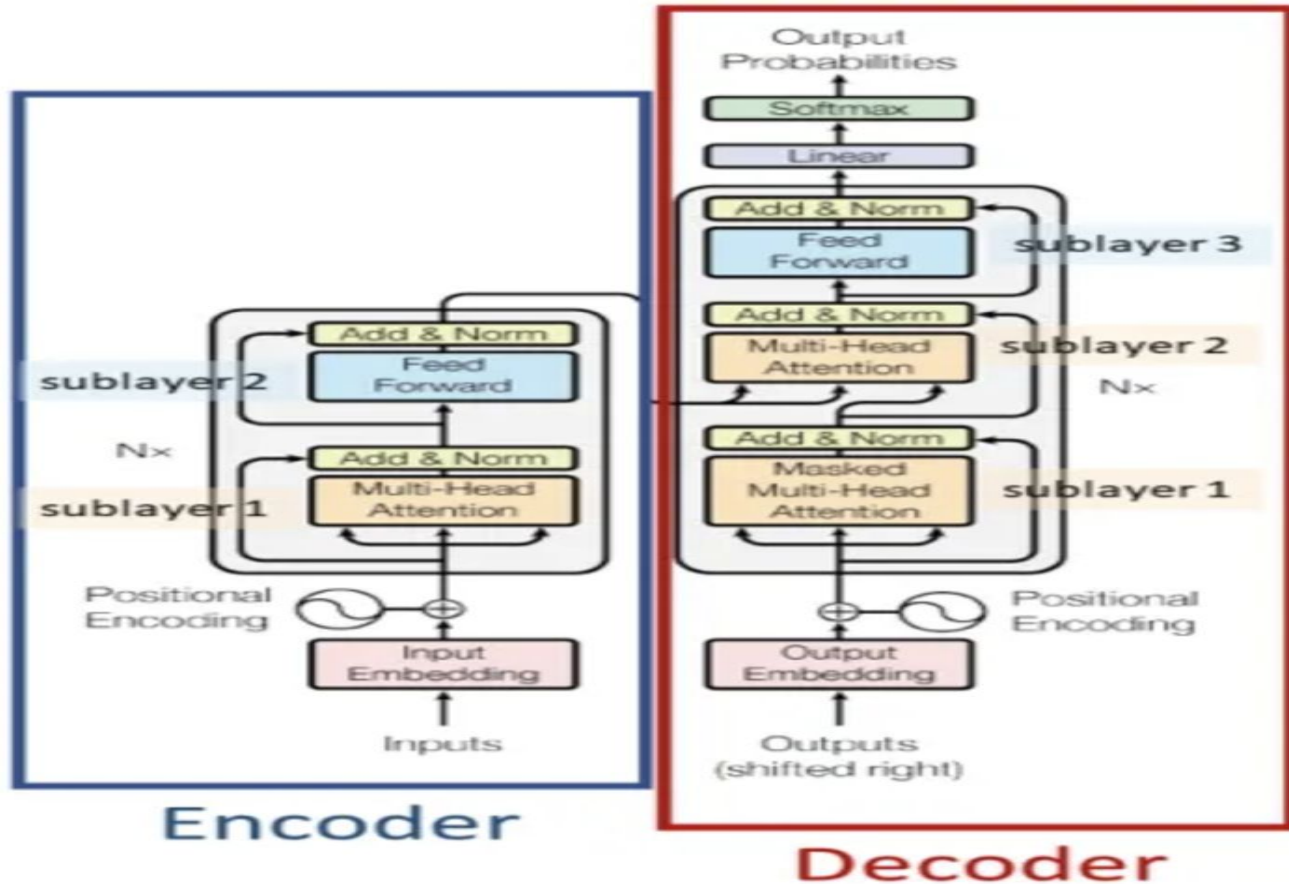
Performance of Word2Vec and TF-IDF

	Accuracy for Document Retrieval using TF-IDF	Accuracy for Document Retrieval using Word2Vec
SQuAD 1.0 Train	0.0141	0.2151
SQuAD 1.0 Validation	0.065	0.1376
SQuAD 2.0 Train	0.0123	0.2018
SQuAD 2.0 Validation	0.0123	0.1072

Reasons for not using RNNs, LSTMs

- RNNs and LSTMs have this core property of **Sequential processing**, whereby sentences must be processed word by word.
 - This is the reason why RNNs and LSTMs can't be trained in parallel. In order to encode the second word in a sentence, we need the previously computed hidden states of the first word, therefore we need to compute that first.
- Another core property is that **Past information is retained through past hidden states**, as in Sequence-to-Sequence models follow the Markov property, each state is assumed to be dependent only on the previously seen state.
 - The point is that the encoding of a specific word is retained only for the next time step, which means that the encoding of a word strongly affects only the representation of the next word, its influence is quickly lost after a few time steps.

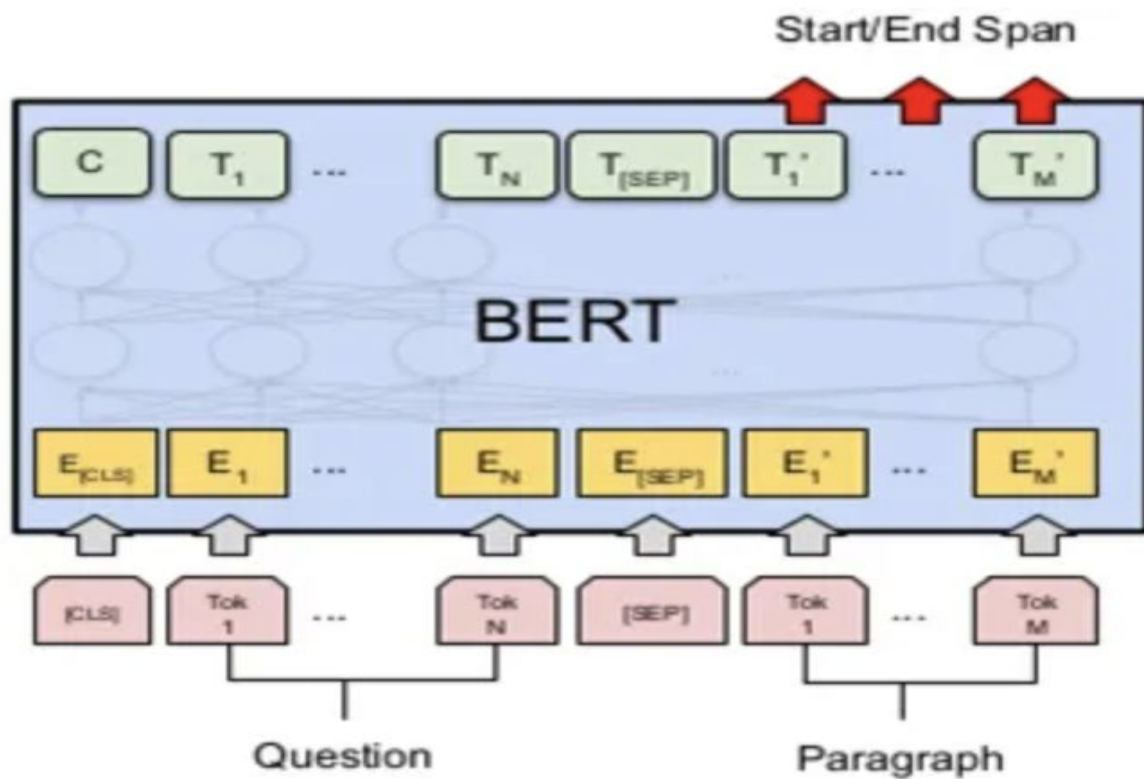
Attention (Transformers) is all you need!



Advantages of Transformers (BERT, specifically!)

- The attention-mechanism looks at an input sequence and decides at each step which other parts of the sequence are important.
 - As an example - when reading a piece of text, we always focus on the word we read but at the same time our mind still holds the important keywords of the text in memory in order to provide context.
- BERT is a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based upon their connection.
 - It is designed to pre-train deep bidirectional representations from an unlabeled text by jointly conditioning on both the left and right contexts.
 - As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks. BERT stacks multiple transformer encoders on top of each other.
 - The transformer is based on the famous multi-head attention module which has shown substantial success in both vision and language tasks.

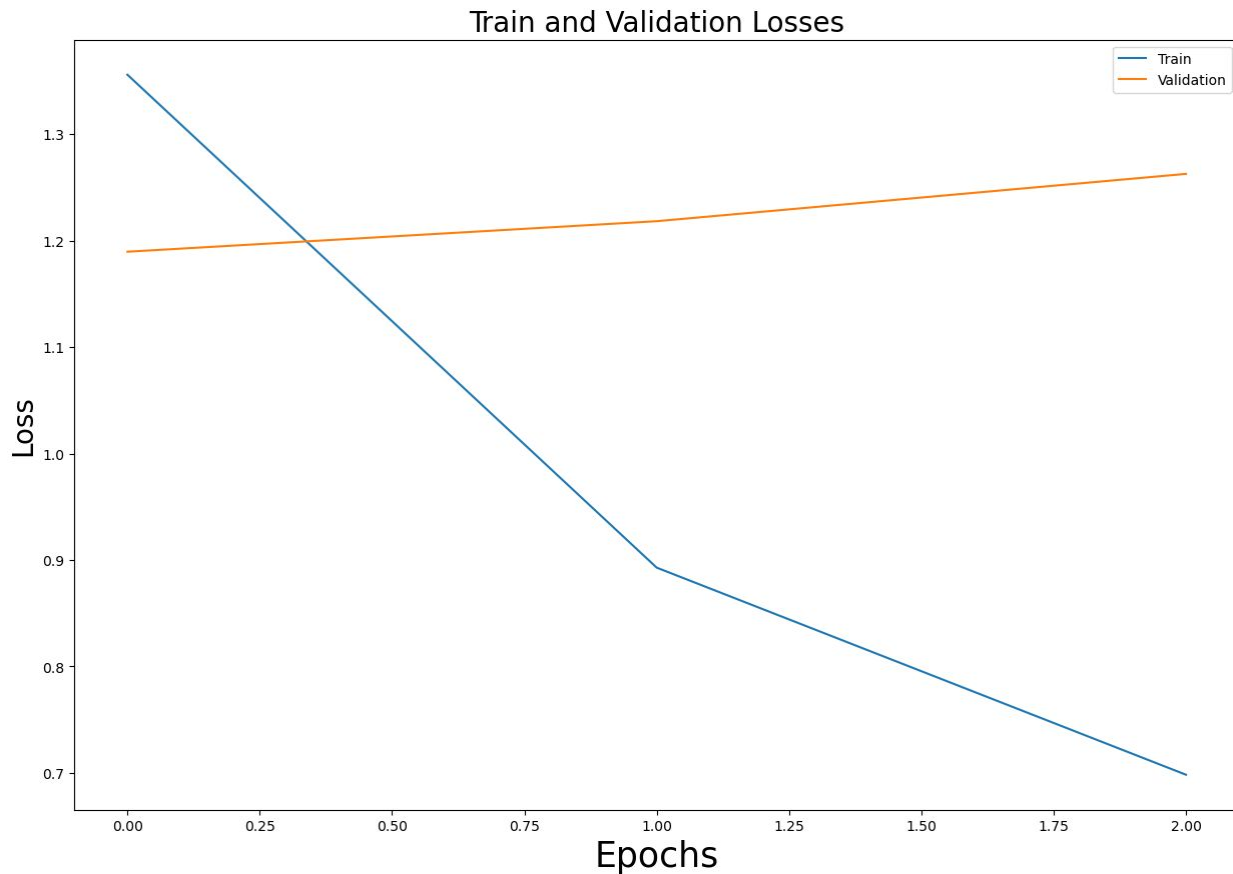
BERT



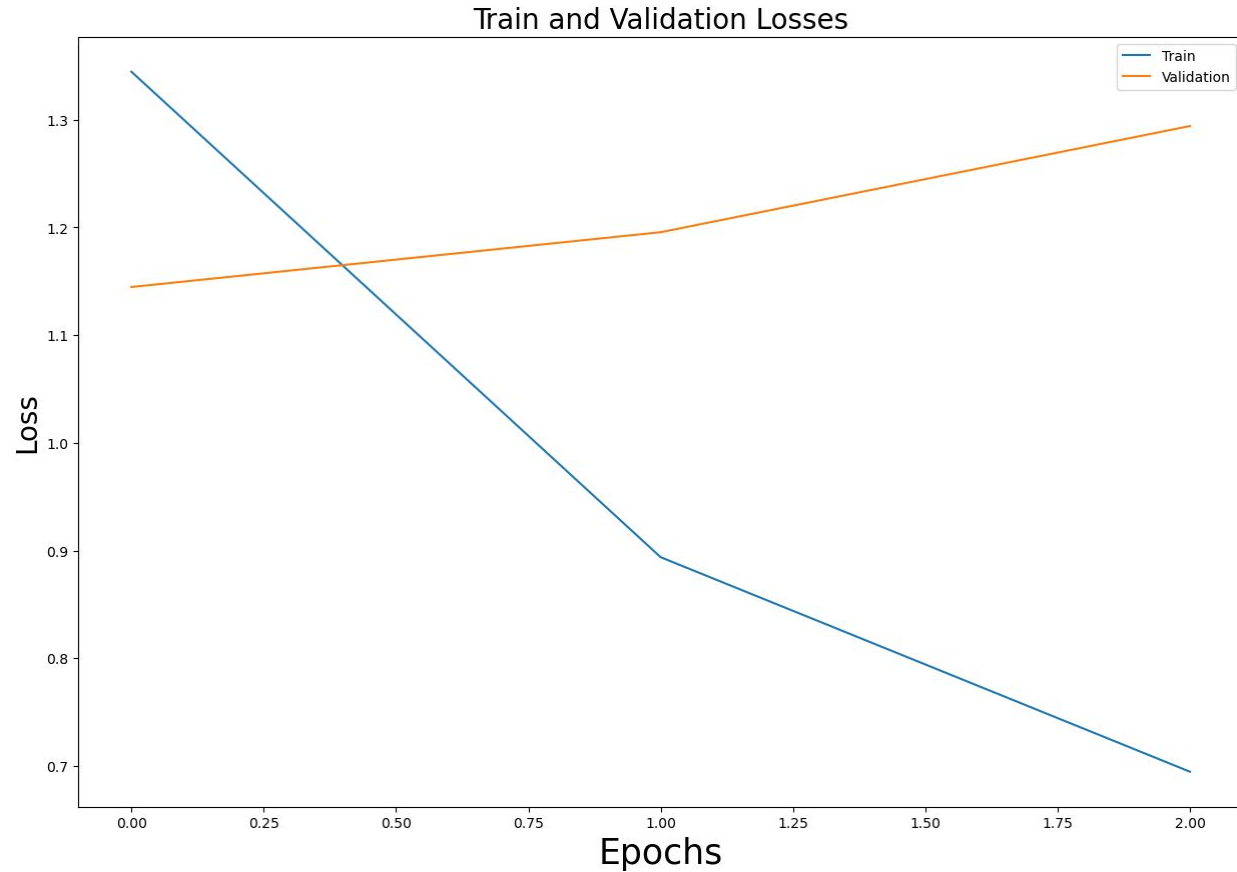
Training Process for fine-tuning our BERT

- **BertForQuestionAnswering** was selected from the **transformers** library, as it suits the current task the most.
 - When a model is instantiated in PyTorch using *from_pretrained()*, the model configuration and pre-trained weights of the specified model are used to initialize the model.
- Tokenization of passages and queries is done using pre-trained tokenizers from **bert-base-uncased**.
- The PyTorch optimizer **AdamW** was used, as it implements **gradient bias correction** as well as **weight decay**.
- **3 epochs** were chosen and the **learning rate** was selected to be 5×10^{-5} .
- A **batch size** of **8** was chosen.
- Time taken for the entire training process was **8.51 hours** on **SQuAD 1.0** and **8.4 hours** for **SQuAD 2.0**.


Training vs Validation Errors wrt Epochs for SQuAD 1.0



Training vs Validation Errors wrt Epochs for SQuAD 2.0



BERT 1 vs bert-large-uncased vs distilbert on SQuAD 1.0 validation data

	Average F1 Score	Average HuggingFace pipeline score
BERT 1	0.2886	0.2444
 bert-large-uncased -whole-word-masking-finetuned-squad	0.79	0.6948
distilbert-base-uncased-distilled-squad	0.7302	0.6607

BERT 2 vs bert-large-uncased vs distilbert on SQuAD 2.0 validation data

	Average F1 Score	Average HuggingFace Pipeline Score
BERT 2	0.3391	0.3013
bert-large-uncased-whole-word-masking-finetuned-squad	0.7704	0.6715
distilbert-base-uncased-distilled-squad	0.7153	0.6423

Reasons why our fine-tuned BERT underperformed?!

- We don't have enough computational resources to run multiple experiments in parallel.
- BERT 1 and BERT 2 have been fine-tuned on only SQuAD 1.0 or SQuAD 2.0 train datasets respectively.
- In reality, not all documents and answers are similar to a question, so domain-specific lexical knowledge is still also necessary for many questions, and the SQuAD datasets don't cover all the cases.

THANK YOU!