CSE4022- Natural Language Processing (E2+TE2)

Review 3

Closed-domain passage based question answering system using NLP and Transfer Learning(BERT)

Team ID: 7

Slot: E2 + TE2

Kaushik SP 20BCE0635

Devansh Sehgal 20BCE0410

Adithiyan Rajan 20BCE0329

Ananya Chembai 20BCE2358

1. Abstract

With this project we aim to develop a comprehensive and robust model for Question Answering with Information Retrieval using Bidirectional Encoder Representations from Transformers (BERT). The BERT architecture, which is a pre-trained natural language processing model, was fine-tuned using transfer learning on a large dataset and next sentence prediction to improve its accuracy in answering questions. We believe that the reported accuracies can be further enhanced through additional hyperparameter tuning. We also discuss the effectiveness of using BERT and transfer learning in natural language processing tasks, and how the approach can be applied to improve the performance of question answering models.

Overall, this research presents a promising approach to developing a robust model for Question Answering with Information Retrieval using BERT and transfer learning. We believe that the findings of this study can help advance the field of natural language processing and contribute to the development of more effective and accurate models for answering questions.

2. Introduction

In today's digital era, where quick and accurate access to information is becoming increasingly crucial, long-form materials such as news pieces, research papers, and technical manuals pose a challenge. To address this, passage-based question answering models have emerged as a viable method of automatically recognizing and extracting relevant information from text. Along with document retrieval systems, these models have gained widespread attention for their ability to process and analyze large amounts of text data, making them crucial tools in the fields of NLP, Information retrieval, and AI. Recently, Transfer Learning techniques have been employed as a promising approach in this area.

Transfer learning is when a model is trained on one single task or domain, and then applied to a new task or domain. In our context, transfer learning can vastly improve performance, and reduce training time by leveraging knowledge gained from large amounts of text data.

Specifically, we are using the Bidirectional Encoder Representations from Transformers (BERT) model, proposed in a paper by Jacob Et al. in 2018. BERT has already shown exceptional performance in multiple use-cases, including question answering, sentiment analysis, and named entity recognition. We are using a dataset called SQuAD. It contains reading comprehension of over a hundred thousand question-answer pairs.

Overall, this project aims to contribute to the growing body of research on passage-based question answering models and help inform future developments in this area. By executing the BERT model, we hope to provide valuable insights into this approach.

3. Literature Review

[1] A system that combines BERT, a state-of-the-art neural network for natural language understanding, with Anserini, an open-source information retrieval toolkit is used [1]. The system can answer natural language questions by retrieving relevant Wikipedia articles and extracting answers from them using BERT. The paper shows that the system outperforms existing open-domain question answering systems on several benchmarks.

[2] proposes a system that improves BERT for open-domain question answering by globally normalizing answer scores across multiple passages retrieved for each question. The system also uses a passage filter to select high-quality passages and a sliding window technique to split long articles into shorter passages. The paper demonstrates that the system achieves state-of-the-art results on four open-domain QA datasets.

The paper [3] describes a system that achieves state-of-the-art results on natural question answering datasets like SQuAD and Natural Questions by using simple techniques such as attention-over-attention, data augmentation and ensembling. The paper also discusses some challenges and limitations of existing natural question answering benchmarks and proposes some directions for future research.

Paper [4] introduces a novel method for improving long-context question answering models by adding a sequence-level objective that encourages them to identify supporting evidence spans for each question. The method uses supervised contrastive learning, which is a technique that maximizes the similarity between positive pairs (question-evidence pairs) and minimizes the similarity between negative pairs (question-non-evidence pairs). The paper shows that this method improves performance on three long-context QA datasets: HotpotQA, Natural Questions Long Answer, and TriviaQA Unfiltered.

[5] introduces a dataset for modeling, understanding, and participating in information seeking dialog. The dataset contains 14K QA dialogs (100K questions in total) that involve two crowd workers: a student who asks a sequence of freeform questions to learn about a hidden Wikipedia text, and a teacher who answers the questions by providing short excerpts from the text. The paper also proposes a dialog-based evaluation metric that measures how well a system can answer questions while taking into account previous questions and answers.

A new baseline model for the Natural Questions dataset, which is a question answering dataset containing queries from google.com and corresponding Wikipedia pages is described in [6]. The model is based on BERT, which is a pre-trained language representation model that can be fine-tuned for various natural language processing tasks. The paper shows that the model significantly improves over previous baselines and reduces the gap between human and machine performance on both long and short answer tasks.

[7] introduces a new pre-training task for BERT, which is a transformer-based language model that can be fine-tuned for various natural language processing tasks. The new task is inspired by reading comprehension and involves selecting an answer span from a relevant passage given a question. The paper shows that this task improves BERT's performance on four reading comprehension benchmarks: SQuAD 1.1, SQuAD 2.0, Natural Questions, and HotpotQA.

A framework for conversational question answering that leverages BERT, a pre-trained language model, to generate contextualized representations of words [8]. The framework also uses a similarity score mechanism to extract only the relevant history turns from previous dialogues that are related to the current question. The paper evaluates the framework on two datasets: CoQA and QuAC, and shows that it outperforms existing methods on both datasets.

[9] explores how generative models for open domain question answering can benefit from retrieving text passages that may contain relevant information. The paper proposes a method that uses a dense retriever to select passages from a large corpus, and then feeds them to a generative model that can attend to them selectively. The paper shows that this method achieves state-of-the-art results on two open domain question answering benchmarks: Natural Questions and TriviaQA.

[10] that introduces a new evaluation suite for retrieval question answering (ReQA) models. ReQA is the task of retrieving a sentence-level answer to a question from an open corpus. MultiReQA consists of eight ReQA tasks from different domains and languages, such as news, trivia, biomedical and customer service. The paper also explores systematic retrieval based evaluation methods for ReQA models and provides baseline results for several existing models.

[11] presents ReQA, an evaluation framework for retrieval question answering (ReQA) models, which comprises two datasets: SQuAD-ReQA and WikiPassageQA-ReQA. ReQA evaluates a model's ability to efficiently retrieve relevant answers from a large set of documents, without relying on document retrieval or answer extraction. The paper establishes baselines for ReQA using both neural encoding models (such as BERT and USE-QA) and classical information retrieval techniques (such as BM25 and TF-IDF). The results show that neural encoding models outperform classical techniques on both datasets, but there is still room for improvement compared to human performance.

Comparison Table:

	Name of the Paper with Year	Methodology/Algorithm:	Challenges	Results
1	End-to-End Open-Domain Question Answering with BERTserini Year: 2019 Authors: Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, Jimmy Lin Journal: arXiv	The paper presents an end-to-end question answering system called BERTserini that combines BERT with the open-source Anserini information retrieval (IR) toolkit. The system operates on a large corpus of Wikipedia articles and uses best practices from IR to identify answers in an end-to-end fashion. BERTserini fine-tunes pre-trained BERT with SQuAD, a dataset for question answering, to achieve high accuracy in identifying answer spans.	One of the challenges of the task is that it is impractical to apply inference exhaustively to all documents in a corpus with current models, mostly based on neural networks. This formulation necessarily requires some type of term-based retrieval technique to restrict the input text under consideration, and hence an architecture quite like the pipelined systems from over a decade ago.	A benchmark test collection shows significant improvements. They demonstrate that BERTserini's architecture simplifies the system. BERTserini is a chatbot that works on laptops and phones. BERT has shown impressive gains in query-based passage reranking, which is relevant to their task. Most question-answering or reading comprehension models use small amounts of text. By combining BERT and Anserini, the paper improves end-to-end question answering by accurately identifying answer spans. BERTserini's simple architecture makes it easy to deploy on different platforms and shows the importance of information retrieval in this field.
2	Multi-passage BERT: A Globally Normalized BERT Model for Open-domain Question Answering	A Multi-passage BERT model for Open-domain Question Answering addresses previous work's incomparable answer scores across passages. The paper proposes global normalisation to normalise	It addresses several unaddressed issues in previous work on Open-domain Question Answering, such as the granularity of passages and the need	It shows that global normalization makes QA models more stable while pinpointing answers from a large number of passages. It also shows that splitting articles into passages with the length of 100

	Year: 2019 Authors: Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, Bing Xiang Journal: arXiv	answer scores across multiple passages, allowing the QA model to use more passages and find better answers. A BERT-based passage ranker selects high-quality passages and splits articles into 100-word passages by sliding window, improving performance by 4. Explicit intersentence matching between questions and passages affects BERT.	for a passage ranker. It also explores the impact of explicit intersentence matching on BERT.	words by sliding window improves performance by 4. The paper demonstrates that leveraging a BERT-based passage ranker gives an additional 2 improvements. Moreover, the paper concludes that explicit intersentence matching is not helpful for BERT. The Multi-passage BERT outperforms all state-of-the-art models on all benchmarks.
3	Frustratingly Easy Natural Question Answering Year: 2019 Authors:Lin Pan, Rishav Chakravarti, Anthony Ferritto, Michael Glass, Alfio Gliozzo, Salim Roukos, Radu Florian, Avirup Sil Journal: arXiv	The framework consists of three steps: (1) retrieving relevant articles using an off-the-shelf search engine, (2) extracting answer candidates using named entity recognition (NER) and noun phrase chunking (NPC), and (3) ranking answer candidates using a pre-trained language model like BERT or RoBERTa. The framework requires no data augmentation or fine-tuning on NQA datasets.	The main challenge addressed in the paper is achieving state-of-the-art performance on the NQ benchmark dataset, which requires models to make an attempt at "knowing what they don't know" by providing a confidence score with each prediction. The paper also addresses the issue of negative instance sub-sampling, which is a severe imbalance in the number of positive to negative (i.e. no answer) spans of text.	A simple transfer learning strategy of fine-tuning the pre-trained BERT model on SQuAD first and then on NQ can nearly match the performance of more complex Attention-over-Attention models. The paper also shows that a simple but effective data augmentation strategy that shuffles the training data outperforms the previous state-of-the-art system trained on 4 million additional synthetically generated QA data. The paper examines several strategies for assembling models to optimise single model performance and notes that small changes in learning rate can have a big impact on F1 scores.

4	Long Context Question Answering via Supervised Contrastive Learning Year: 2022 Authors: Avi Caciularu, Ido Dagan, Jacob Goldberger, Arman Cohan Journal: arXiv	A supervised contrastive learning for long-context QA tasks. An evidence selector and answer predictor comprise the framework. The evidence selector finds sentences from a long document or multiple documents that support the question. Based on selected evidence spans, the answer predictor predicts. The contrastive loss function encourages relevant evidence spans and discourages irrelevant ones.	Noise in long documents can confuse the model. Long documents may exceed the input limit of pre-trained models and require chunking or pruning, which may lose important information. Finally, complex questions require multiple reasoning steps and cross-document inference, which can challenge the model's accuracy and coherence.	As shown in the table provided in the paper, the addition of the authors' LQE to the similar approaches displays an improvement in performance across all examined models and benchmarks, demonstrating its benefit. Further, maximizing the question-evidence similarity resulted also in evidence detection improvement.
5	QuAC : Question Answering in Context Year: 2018 Authors: Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, Luke Zettlemoyer Journal: arXiv	The paper proposes a neural network model that consists of three components: a bi-directional attention flow (BiDAF) encoder, a dialog context encoder, and an answer pointer. The BiDAF encoder uses co-attention to encode both the passage and the question into contextualized representations. The dialog context encoder uses RNNs to encode the dialog history, which consists of previous	QA challenges like context-dependence, coreference resolution, ellipsis resolution, unanswerability detection, and follow-up question generation. Questions are often context-dependent. Coreference resolution is needed to resolve pronouns and other references to their antecedents in questions. . Unanswerability	The paper reports results on the QuAC dataset, which contains 14K information-seeking QA dialogs (100K questions in total). The dialogs are collected from Wikipedia articles on various topics such as history, science, politics, etc. The paper shows that its model outperforms several baselines on QuAC metrics, such as F1 score, human accuracy equivalence (HAE), and dialog act F1. F1 score measures how well the model predicts answer spans compared to human annotations. HAE measures how well the model matches human

		questions and answers in the conversation. The answer pointer uses another RNN to predict an answer span from the passage or a special token indicating unanswerability or follow-up questions. The model is trained with cross-entropy loss on answer span prediction and dialog act prediction.	detection means passage questions may be out of scope or unanswerable. Based on previous answers, follow-up questions may request additional information.	performance on answering questions correctly or indicating unanswerability correctly. Dialog act F1 measures how well the model predicts dialog acts such as affirmation, negation, follow-up question request.
6	A BERT Baseline for the Natural Questions Year: 2019 Authors: Chris Alberti, Kenton Lee, Michael Collins Journal: arXiv	A new baseline for the large-scale question answering (QA) dataset Natural Questions (NQ), which contains Google Search user queries. The paper adds an answer type classifier, sliding windows over long documents, and joint training on long and short answer tasks to BERT. The paper trains and infers without additional data or resources.	Handling questions with no answer or multiple answers, dealing with long documents that may contain irrelevant information, and requiring fine-grained understanding of natural language queries.	The paper reports results on NQ using both development and test sets. The paper shows that its BERT-based system achieves state-of-the-art performance on NQ, surpassing previous baselines by a large margin. The paper also shows that its system reduces the gap between model F1 scores and human upper bound by 30% and 50% relative for long and short answer tasks respectively.
7	Span Selection Pre-training for Question Answering Year: 2019	BERT is trained to select answer spans from passages using span selection pre-training (SSPT) on large unlabeled text corpora. Masking named entities or noun phrases from sentences creates cloze-like training instances, which BERT	The paper lists several QA challenges, including long documents with multiple answer candidates, questions that have no answer or require reasoning, and domain	The paper reports results on four QA datasets: SQuAD 2.0, NQ, TriviaQA, and HotpotQA. The paper shows that its SSPT method improves QA performance on these datasets over both BERT-BASE and BERT-LARGE models. The paper also shows that its method achieves state-of-the-art results

	Authors: Michael Glass, Alfio Gliozzo, Rishav Chakravarti, Anthony Ferritto, Lin Pan, G P Shrivatsa Bhargav, Dinesh Garg, Avirup Sil Journal: arXiv	predicts from their contexts. A span selection loss function in the paper encourages BERT to choose coherent and informative spans as answers.	knowledge or common sense. Relying on artificial cloze questions or generating answers from model parameters rather than passages	on SQuAD 2.0 and NQ among single-model systems.
8	BERT-CoQAC: BERT-based Conversational Question Answering in Context Year: 2021 Authors: Munazza Zaib, Dai Hoang Tran, Subhash Sagar, Adnan Mahmood, Wei E. Zhang, Quan Z. Sheng Journal: arXiv	BERT-CoQAC uses BERT to incorporate conversational history into context-based question answering. Context-query relevance selects relevant history turns, and then history questions and embeddings model the history conversation to improve answers.	The paper mentions some challenges for conversational question answering in context, such as how to select relevant history turns, how to model history conversation effectively, and how to evaluate the system performance.	The paper reports experimental results on two datasets: CoQA and QuAC. The paper shows that BERT-CoQAC outperforms several baselines and achieves state-of-the-art results on both datasets. The paper also conducts ablation studies and error analysis to demonstrate the effectiveness of each component of BERT-CoQAC and identify some limitations and future directions.
9	MultiReQA: A Cross-Domain Evaluation for Retrieval Question Answering Models	The paper proposes a new evaluation suite for retrieval question answering (ReQA) models that consists of eight ReQA tasks from different	The paper mentions several challenges for ReQA, such as dealing with diverse question types and answer formats, handling	The paper evaluates its baseline models on MultiReQA using ReQA-MRR and shows that they perform poorly on most tasks, especially on those that require complex reasoning or

	Year: 2020 Authors: Mandy Guo, Yinfei Yang, Daniel Cer, Qinlan Shen, Noah Constant Journal: arXiv	domains and sources. The paper also proposes a new metric called ReQA-MRR that measures the mean reciprocal rank of the first correct answer sentence retrieved by a model. The paper also introduces two baseline models for ReQA: BM25 and BERTserini.	domain-specific terminology and knowledge, and evaluating model performance across different tasks and domains.	domain knowledge. The paper also conducts qualitative analysis and shows that the baseline models often fail to retrieve relevant answer sentences due to lexical mismatch, syntactic variation, or semantic ambiguity.
10	ReQA: An Evaluation for End-to-End Answer Retrieval Models Year: 2019 Authors:Amin Ahmad, Noah Constant, Yinfei Yang, Daniel Cer Journal: arXiv	The paper introduces Retrieval Question-Answering (ReQA), a benchmark for evaluating large-scale sentence-level answer retrieval models. The paper also proposes two neural encoding models based on BERT and USE-QA, which encode questions and answers into dense vectors and compute their similarity using dot product. The paper compares these models with classical information retrieval techniques such as BM25 and TF-IDF.	The paper identifies four challenges for answer retrieval models: 1) They should be able to handle natural language variations in questions and answers; 2) They should be robust to noise and irrelevant information in large-scale corpora; 3) They should be efficient and scalable to handle millions of candidate answers; 4) They should be context-aware, meaning that they should consider the surrounding sentences when selecting an answer.	The paper reports experimental results on two datasets: SQuAD-ReQA, which is derived from SQuAD v1.1, and NaturalQuestions-ReQA, which is derived from Natural Questions. The paper shows that neural encoding models outperform classical information retrieval techniques on both datasets, especially on NaturalQuestions-ReQA, which has more diverse and complex questions and answers. The paper also shows that BERT-based models perform better than USE-QA-based models on both datasets.

4. Proposed System and Explanation

Here we describe in detail the steps taken to achieve the project's objective. This procedure will utilize the Knowledge discovery in datasets (KDD) method, as it is primarily a scientific endeavor. This work is implemented using the KDD phases of Selection, Preprocessing Data, Transformation, Data Mining, and Interpretations/Evaluation. The majority of these steps are performed using Python. The supporting libraries and resources for the application's development are pandas, numpy, seaborn gensim, matplotlib, and scikit-learn.

A. Dataset Selection

The dataset used to construct the question and answer system was compiled by and is referred to as the Stanford Question Answering Dataset (SQuAD). It was generated by crowdworkers and contains over one hundred thousand question-answer pairs based on over five hundred articles for reading comprehension. It is a dataset with a closed domain, so every answer is a portion of text from the associated passage.

B. Pre-Processing

The dataset is written in natural language processing (English) and must be pre-processed before it can be examined and utilised. This involved examining the structure of the dataset to determine its characteristics and attributes.

The datasets (training and dev) are untangled and converted to a clean dataframe.

There were no null values in the dataset, so no action was required to handle null values. The dataset was analysed using Word Embeddings to identify words with similar meanings.

The required cleaning process prior to word embeddings included converting words to lowercase for uniformity, removing punctuation, stopwords, and other diacritics, as well as white space and redundant columns.

In this project, pre-trained transformer models were created using the simple transformers library from huggingface transformers[45], and for question answering in simple transformers, the data must be preprocessed into python list dictionaries. Each dictionary possesses the "context" and "qas" (list of questions and answers) attributes, with the "qas" attribute containing the id, a question, and a list of correct answers.

C. Exploratory Data Analysis

Any data analysis problem relies heavily on exploratory data analysis, which is frequently performed visually. Typically, it is performed to gain useful insights about the dataset.

1. Wordcloud

Wordcloud, also known as a tag cloud, is a text-based data visualisation that displays the most frequent and significant words within a dataset. Significant words are written in larger, bolder fonts, while less significant words are displayed in smaller, lighter fonts or are omitted entirely.

The advantage of the Wordcloud is that it provides a quick overview of the text you are analysing, which aids in the comprehension of the dataset. This project's Wordcloud was generated in Python using the pandas, matplotlib, and Wordcloud packages.

2. Word Embeddings
Creating word vectors is the process of taking a large corpus of text and creating a vector for each word such that words that share common contexts in the corpus are located in close proximity to one another in the vector space.
These word vectors can get amazingly good at capturing contextual relationships between words (example vectors for black, white and red would be close together) and we get far better performance with using these vectors instead of raw words for NLP tasks like text classification or new text generation. We use Word2Vec for this process.

3. Word2vec
Word2vec is a neural network-based technique for word embedding that uses an input layer, a hidden layer, and an output layer to model words as vectors.
Using the paragraphs as input, the output layer contains the words as vectors. It has two methods for implementing embedding training: continuous-bag-of-words (CBOW) and skip-gram. Word2vec employs the genism open-source Python library to implement word embedding. A similarity metric was utilised to examine the proximity of words.

D. Application of Question answering System
A question-answering system has a wide range of applications and numerous potential use cases. One of the most prevalent uses of (QA) systems is in the context of Frequently Asked Questions (FAQ), where a system is designed to provide quick and accurate responses to frequently asked questions. This can be especially helpful for customer service departments, technical support teams, and other organisations that receive a large volume of questions with similar answers. Additionally, QA systems can be used as chatbots that interact with users in a conversational manner and provide pertinent information in response to user queries. This is particularly useful for businesses and organisations that want to provide customer support around-the-clock without requiring a large staff of human operators. A common application of QA systems is as a web search engine, such as Google or Bing. In this context, the system is intended to provide answers to user queries based on a vast array of available information, such as web pages, articles, and other resources. QA systems can also be used for educational purposes, giving students access to a large knowledge base and permitting them to ask questions and receive answers in real time. Overall, the adaptability and potential applications of QA systems make them a promising area for academic and industrial research and development. Specifically,

Education: Closed domain question answering systems can help students learn from educational

materials that cover a particular subject, such as history, biology, or mathematics123. They can also help teachers assess students' knowledge and provide feedback2.
Medicine: Closed domain question answering systems can help doctors diagnose patients, prescribe treatments, or access medical literature that are specific to a certain disease, symptom, or drug45. They can also help patients find health information or advice that are relevant to their condition.
Law: Closed domain question answering systems can help lawyers find legal information or precedents that are related to a certain case, statute, or regulation45. They can also help citizens understand their rights and obligations under the law4.
Engineering: Closed domain question answering systems can help engineers design, test, or troubleshoot systems or devices that are based on a certain technology, standard, or protocol45. They can also help users operate or maintain such systems or devices4.

E. Model

Transfer learning is an important machine learning technique that transfers information from one task to a new, similar activity in order to either reduce the need for fine-tuning, processing time, or improve the model's accuracy.

Transfer learning is frequently used to solve NLP challenges, like as question answering, especially when pre-trained Transformer models are employed. Transformer is a neural network design constructed employing a self-attention mechanism that functions exceptionally well for language comprehension. Transformer models are managed in this project using the Simple Transformers library.

For each NLP task, a class exists in simple transformers, and training, evaluation, and prediction are performed using objects of this class. The Question Answering Model object is formed and its hyperparameters for fine-tuning. The model type and model name parameters are required, while the args parameter is optional; if not provided, default values are used.

Simple transformers support the following model types: BERT, ELECTRA, Longformer, XLM, ALBERT, DistilBERT, MobileBERT, XLM-RoBERTa, XLNet, and RoBERT. In this project, the BERT model is employed for the question-answering task.

The model name defines the required architecture and trained weights. The term "bert-based-cased" is used. The 'bertbased-cased' is a Huggingface Transformers-compatible, self-supervised model that was trained on the English language utilising masked language modelling and next sentence prediction. BERT has yielded positive results for NLP-related tasks throughout time, and it is being employed by Google's search engine.

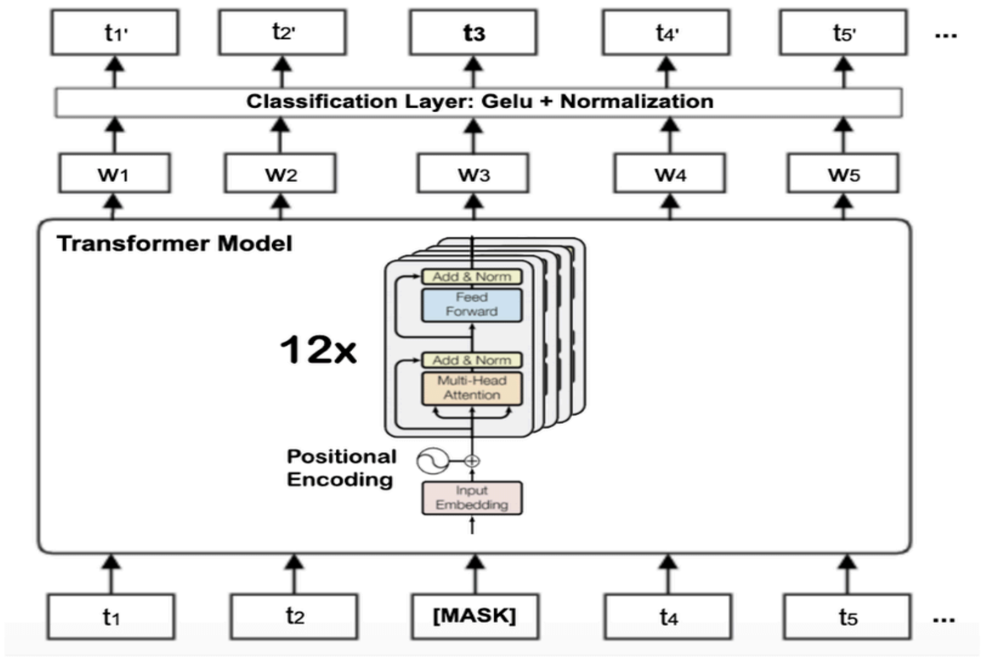
Comparing BERT embeddings to Word embeddings created with Word2Vec yields various embeddings for the same words based on their context-dependent meaning.

F. Implementation
Implementation utilises the Python programming language and the Google Colab runtime environment. The transformers package has a BERT user interface.

To train the model, a random sample of 5,000 records is extracted from the train dataset. In addition, a random sample of 1,000 is picked from the remaining train dataset to test the model. System constraints necessitated a random sampling procedure.

4. 1 Architecture Diagram:

BERT Architecture



The user inputted text is broken down into tokens and delimiters by **[CLS]** and **[SEP]** tokens.
[CLS] tokens are added to the beginning of the Questions
[SEP] tokens are added at the end of the Questions and paragraphs.

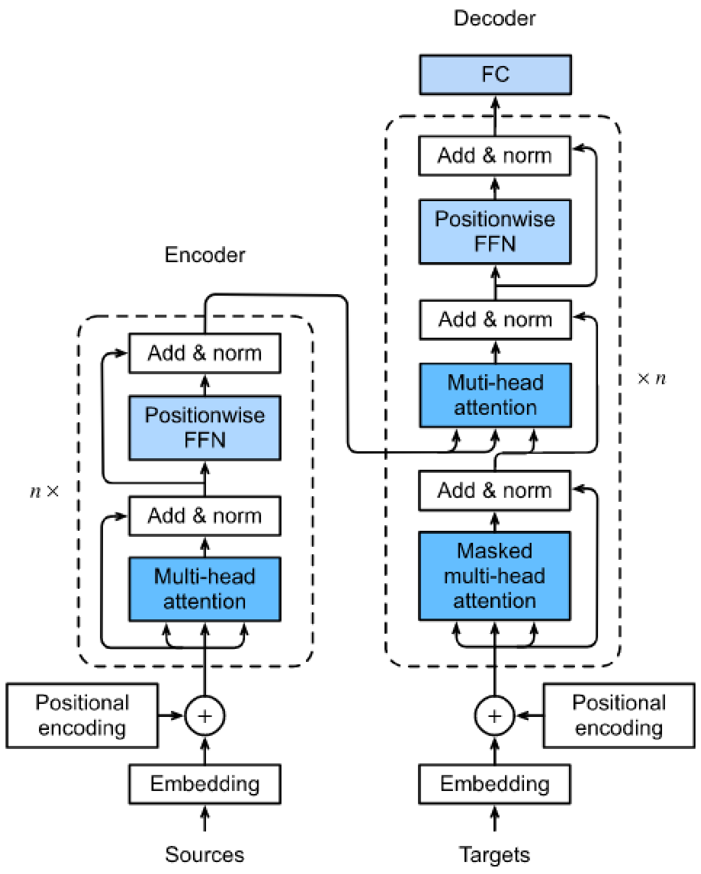
The delimited text is then given a numerical representation called **Input Embedding**. This is found by using the Token Embedding (which is the tokenized form of the delimited text) and the Segment Embedding. **Token Embeddings** represent the meaning of individual words, which helps understand the relationship between different words. **Segment Embeddings** represent which sentence the input text belongs to, which helps understand the structure of text.

Input embeddings = Token Embedding + Segment Embedding

Now we pass input embeddings through layers of transformers, to generate **Word Embedding** and **Start Vector**.
Probability of each word being the Start Word is calculated by taking the dot product between the Final Embedding of the word and the Start Vector and converting those scores into probabilities by using the Soft Max function.

The process is repeated to generate the next consecutive words until the End Word. Once the End Word and Start Word, we can generate a full answer text from the context by grabbing the Start Word to End Word.

Transformer Architecture



Input Encoding: The input sequence is first encoded using an embedding layer, which maps each token in the sequence to a high-dimensional vector representation. The embedding layer helps the model to understand the meaning of each word in the sequence.

Positional Encoding: In addition to the token embeddings, the input sequence is also encoded with positional information. This is done using a positional encoding function, which adds a fixed vector to each token embedding that indicates the position of the token in the sequence.

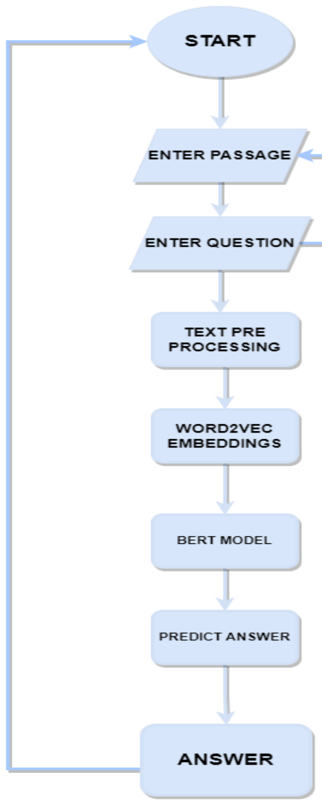
Multi-Head Attention: The Transformer architecture uses multi-head attention, which allows the model to attend to different parts of the input sequence simultaneously. The input sequence is split into multiple heads, each of which is processed independently. Within each head, attention weights are computed between each token in the sequence and all other tokens in the sequence. The attention weights indicate how much attention should be paid to each token when computing the output for that head.

Feedforward Network: After the multi-head attention layer, the output is passed through a feedforward network, which applies a linear transformation followed by a non-linear activation function. This helps the model to learn more complex patterns in the input sequence.

Decoder: The decoder is similar to the encoder, but it also includes an additional multi-head attention layer that attends to the output of the encoder. This allows the decoder to incorporate information from the input sequence when generating its output.

Softmax Layer: The final layer of the model is a softmax layer, which maps the output of the decoder to a probability distribution over the possible output tokens. The token with the highest probability is selected as the model's output.

4.2 Flow Diagram:



The user is prompted to enter a passage of his choice and a question relevant to that passage. The model then performs various operations on that text to produce the desired output.

Text preprocessing such as lower casing, removal of punctuation, removal of extra white spaces, removal of stop words, stemming and lemmatization is performed on the text. Text preprocessing is necessary to clean and transform raw textual data into a format that can be easily understood and analyzed by machine learning algorithms.

Next, **Word2Vec Embeddings** are produced. Word2Vec is a shallow, two-layer neural network that takes a large corpus of text as input and produces a vector space, where each unique word in the corpus is assigned a corresponding vector in the space. It is used to learn word representations that capture the semantic and syntactic relationships between words.

The **Bert Model** then processes the clean text and produces the output as an **answer** to the question asked by the user.

4.3 Pseudocode

Text preprocessing and word embeddings

```
Function CleanText(text):
    Input: The raw text to be cleaned (text)
    Output: The cleaned text
    Convert text to lowercase;
    Remove punctuation from text;
    Remove extra spaces from text;
    Remove stop words from text;
    Stem text;
    Lemmatize text;
    return text;
```

```
Function Embed(text, model):
    Input: The text to be embedded (text) and the pre-trained
           Word2Vec model (model)
    Output: The resulting embeddings
    tokens $\leftarrow$ tokenize text;
    embeddings $\leftarrow$ empty list;
    for token $\in$ tokens do
        if token in model then
            embeddings.append(model[token]);
    return embeddings;
```

Training model

```
Function TrainBERT($D$, $M$, $O$, $S$, $T$):
    Input: The SQuAD dataset ($D$), the pre-trained BERT model
           ($M$), the optimizer ($O$), the learning rate scheduler ($S$), and
           the tokenization method ($T$)
    Output: The trained BERT model
    for $epoch \leftarrow 1$ to $N$ do
        for $(q, c, a) \in D$ do
            $q \leftarrow$ CleanText ($q$);
            $c \leftarrow$ CleanText ($c$);
            $q\_tokens \leftarrow$ Tokenize ($q$, $T$);
            $c\_tokens \leftarrow$ Tokenize ($c$, $T$);
            $input \leftarrow$ Convert $q\_tokens$ and $c\_tokens$ to BERT input
              format;
            $output \leftarrow M(input)$;
            $loss \leftarrow$ Compute loss between $output$ and $a$;
            $loss$.backward();
            $O$.step();
            $S$.step();
    return $M$;
```

This pseudocode shows how a Natural Language Processing (NLP) project's BERT model is preprocessed and trained. The main steps include text cleaning, embedding, and training the BERT model. Let's break down each part of the pseudocode:

1. Text Cleaning (CleanText Function): This function takes raw text as input and performs several preprocessing steps to clean it:
    - Convert the text to lowercase
    - Remove punctuation
    - Remove extra spaces
    - Remove stop words (common words that do not carry much meaning)
    - Apply stemming (reducing words to their root form)
    - Apply lemmatization (converting words to their base form)

      The cleaned text is then returned as output.
2. Embedding (Embed Function): This function takes text and a pre-trained Word2Vec model as input and generates word embeddings for the text. The process involves:

        - Tokenizing the text into individual words (tokens)
        - Initializing an empty list for embeddings
        - Iterating through the tokens and, if the token exists in the pre-trained model, appending the corresponding embedding to the list
          The function then returns the list of embeddings.
3. Training BERT (TrainBERT Function): This function takes the SQuAD dataset, a pre-trained BERT model, an optimizer, a learning rate scheduler, and an embedding method as input. It trains the BERT model using the following steps:
    - Loop through a specified number of epochs
        - For each question, context, and answer triplet in the dataset:
            - Clean the question and context text using the CleanText function
            - Generate embeddings for the question and context using the Embed function
            - Convert the embeddings to BERT input format
            - Pass the input through the BERT model to get the output
            - Compute the loss between the model's output and the actual answer
            - Perform backpropagation to update the model's parameters
            - Update the optimizer and learning rate scheduler
              The trained BERT model is returned as output.

5. Experiment and Results

5.1 Dataset

The Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset consisting of questions posed by crowdworkers on a set of Wikipedia articles. The answer to every question is a segment of text, or span, from the corresponding reading passage. There are 100,000+ question-answer pairs on 500+ articles.

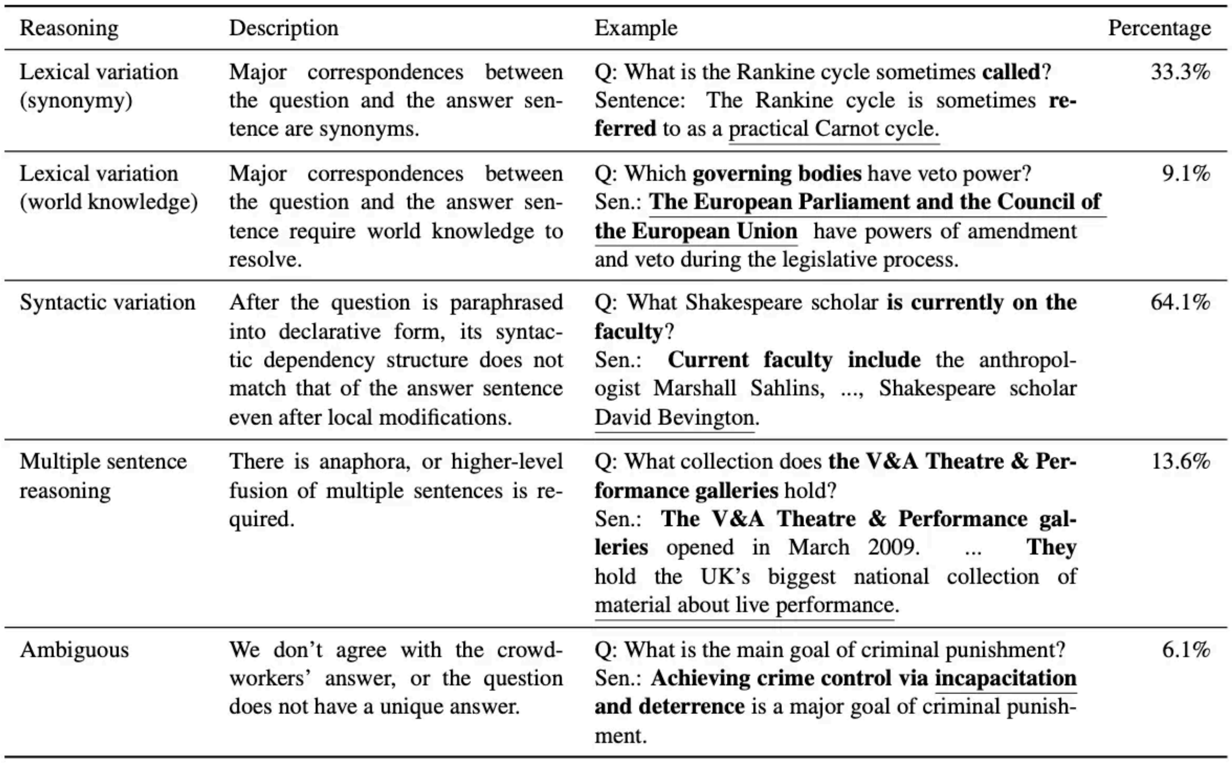
SQuAD focuses on the task of question answering. It tests a model’s ability to read a passage of text and then answer questions about it.

Data fields
- id: tensor containing article id.
- title: tensor containing the title.
- context: tensor containing a context from the article.
- question: tensor containing the question
- text: tensor containing the text
- answer_start: tensor that contains the starting index of the answer.

- is_impossible: A label that represents whether answering is possible or impossible. If it is impossible, the label represents the value 1 for True. If not impossible, the label represents the value o for False.

Properties of (SQuAD)

- It was found that dates and numbers made up 19.8% of the answers, nouns made up 32.6%, noun-phrases made up 31.8%, and other categories made up the remaining 15.8%.
- The dataset was found to have a diverse range of syntactic divergence.
- The creators sampled questions from the development set and manually labeled questions into different categories of reasoning required to answer them. Thus the answers require a rearrangement of words to find the answer



Reasoning	Description	Example	Percentage
Lexical variation (synonymy)	Major correspondences between the question and the answer sentence are synonyms.	Q: What is the Rankine cycle sometimes **called**? Sentence: The Rankine cycle is sometimes **referred** to as a practical Carnot cycle.	33.3%
Lexical variation (world knowledge)	Major correspondences between the question and the answer sentence require world knowledge to resolve.	Q: Which **governing bodies** have veto power? Sen.: **The European Parliament and the Council of the European Union** have powers of amendment and veto during the legislative process.	9.1%
Syntactic variation	After the question is paraphrased into declarative form, its syntactic dependency structure does not match that of the answer sentence even after local modifications.	Q: What Shakespeare scholar **is currently on the faculty**? Sen.: **Current faculty include** the anthropologist Marshall Sahlins, ..., Shakespeare scholar David Bevington.	64.1%
Multiple sentence reasoning	There is anaphora, or higher-level fusion of multiple sentences is required.	Q: What collection does **the V&A Theatre & Performance galleries** hold? Sen.: **The V&A Theatre & Performance galleries** opened in March 2009. ... **They** hold the UK's biggest national collection of material about live performance.	13.6%
Ambiguous	We don't agree with the crowd-workers' answer, or the question does not have a unique answer.	Q: What is the main goal of criminal punishment? Sen.: **Achieving crime control via incapacitation and deterrence** is a major goal of criminal punishment.	6.1%

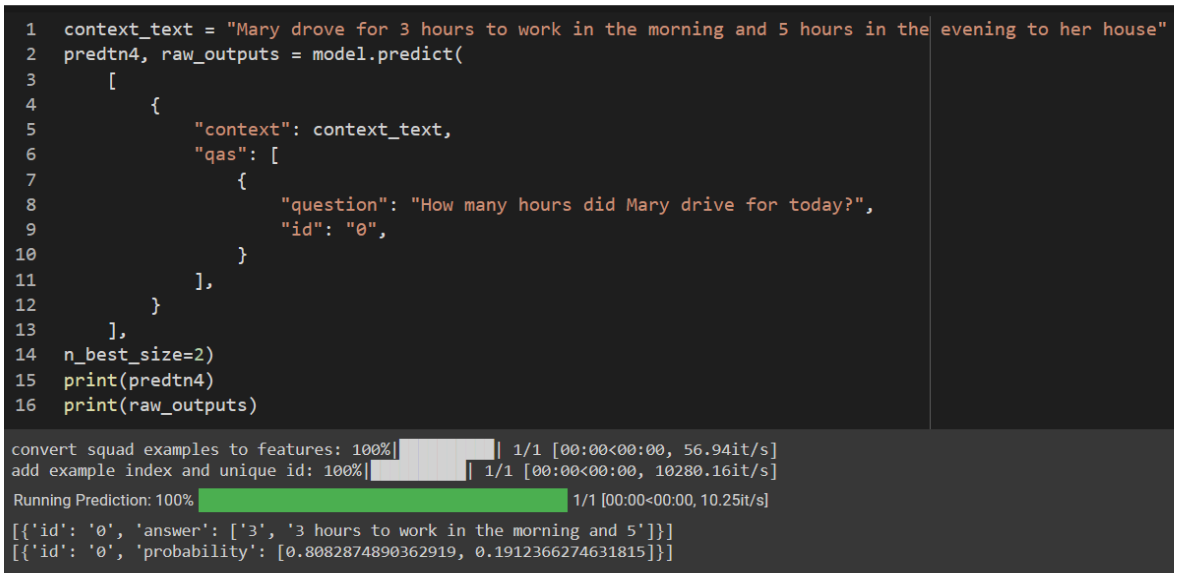
Advantages of SQuAD

- SQuAD is a large dataset - Other reading comprehension datasets such as MCTest and Deep Read are too small for complex models. MCTest only has a total of 2,640 questions, Deep Read a total of 600 questions while SQuAD has a whopping 100,000+ questions.

- SQuAD is challenging - In other similar datasets, the answer to a given question occurs in multiple documents. In SQuAD, however, the model only has access to a single passage, presenting a much more difficult task since it isn't as forgiving to miss the answer.
- SQuAD requires reasoning. - its answers are more complex and thus require more-intensive reasoning, thus making SQuAD better for evaluating model understanding and capabilities.

5.1.1 Methodology with Dataset

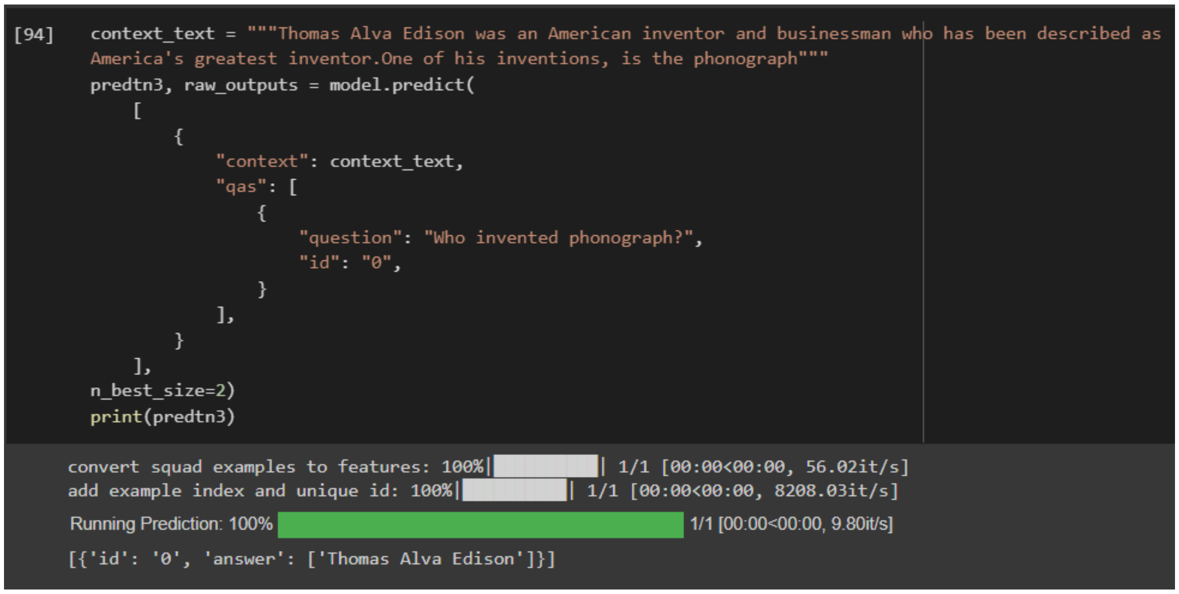
Example 1:



```
context_text = "Mary drove for 3 hours to work in the morning and 5 hours in the evening to her house"
predtn4, raw_outputs = model.predict(
    [
        {
            "context": context_text,
            "qas": [
                {
                    "question": "How many hours did Mary drive for today?",
                    "id": "0",
                }
            ],
        }
    ],
n_best_size=2)
print(predtn4)
print(raw_outputs)
```

```
convert squad examples to features: 100%|██████████| 1/1 [00:00<00:00, 56.94it/s]
add example index and unique id: 100%|██████████| 1/1 [00:00<00:00, 10280.16it/s]
Running Prediction: 100% ████████████████████ 1/1 [00:00<00:00, 10.25it/s]
[{'id': '0', 'answer': ['3', '3 hours to work in the morning and 5']}]
[{'id': '0', 'probability': [0.8082874890362919, 0.1912366274631815]}]
```

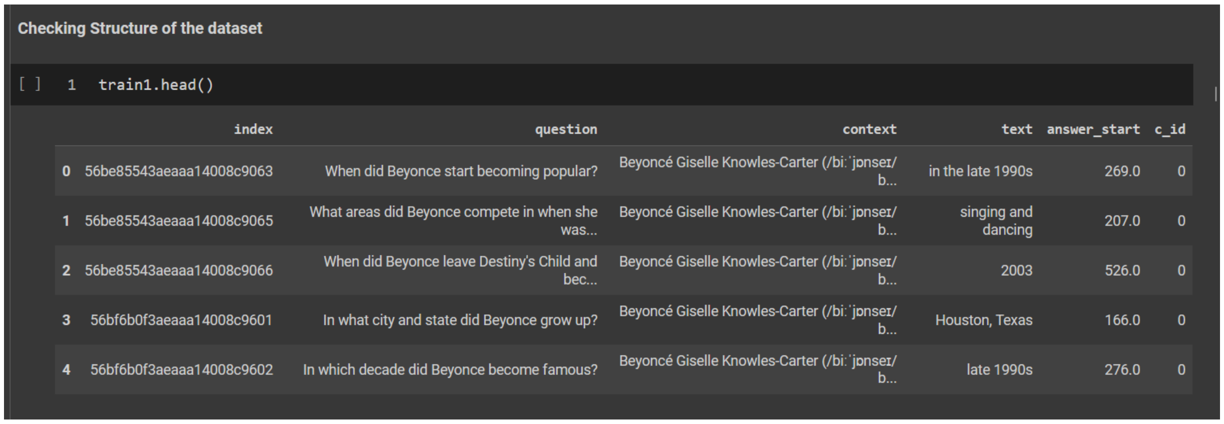
Example 2:



```
[94]  context_text = """Thomas Alva Edison was an American inventor and businessman who has been described as
      America's greatest inventor.One of his inventions, is the phonograph"""
      predtn3, raw_outputs = model.predict(
          [
              {
                  "context": context_text,
                  "qas": [
                      {
                          "question": "Who invented phonograph?",
                          "id": "0",
                      }
                  ],
              }
          ],
      n_best_size=2)
      print(predtn3)
```

```
convert squad examples to features: 100%|██████████| 1/1 [00:00<00:00, 56.02it/s]
add example index and unique id: 100%|██████████| 1/1 [00:00<00:00, 8208.03it/s]
Running Prediction: 100% ████████████████████ 1/1 [00:00<00:00, 9.80it/s]
[{'id': '0', 'answer': ['Thomas Alva Edison']}]
```

Dataset Sample



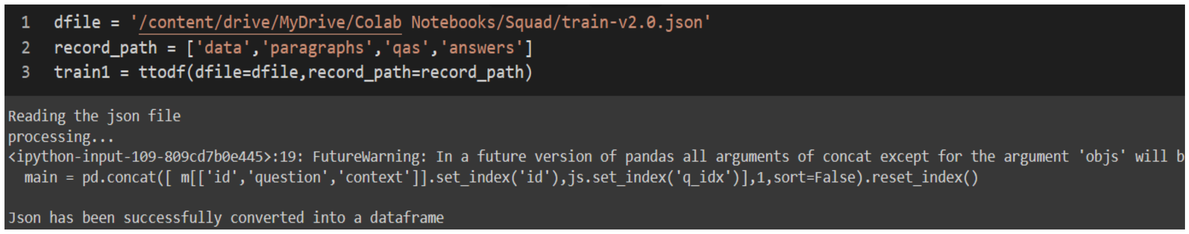
Checking Structure of the dataset

```
train1.head()
```

	index	question	context	text	answer_start	c_id
0	56be85543aeaaa14008c9063	When did Beyonce start becoming popular?	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	in the late 1990s	269.0	0
1	56be85543aeaaa14008c9065	What areas did Beyonce compete in when she was...	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	singing and dancing	207.0	0
2	56be85543aeaaa14008c9066	When did Beyonce leave Destiny's Child and bec...	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	2003	526.0	0
3	56bf6b0f3aeaaa14008c9601	In what city and state did Beyonce grow up?	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	Houston, Texas	166.0	0
4	56bf6b0f3aeaaa14008c9602	In which decade did Beyonce become famous?	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	late 1990s	276.0	0

Text PreProcessing Steps:

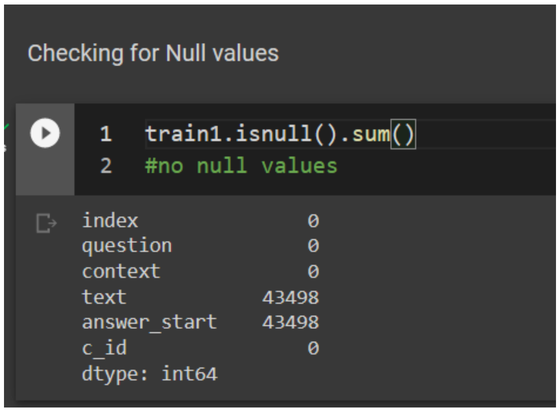
1) Conversion of convoluted json structure to clean pandas dataframe format.



```
dfile = '/content/drive/MyDrive/Colab Notebooks/Squad/train-v2.0.json'
record_path = ['data','paragraphs','qas','answers']
train1 = ttodf(dfile=dfile,record_path=record_path)
```

```
Reading the json file
processing...
<ipython-input-109-809cd7b0e445>:19: FutureWarning: In a future version of pandas all arguments of concat except for the argument 'objs' will b
  main = pd.concat([ m[['id','question','context']].set_index('id'),js.set_index('q_idx')],1,sort=False).reset_index()

Json has been successfully converted into a dataframe
```

2) Check and resolve null values

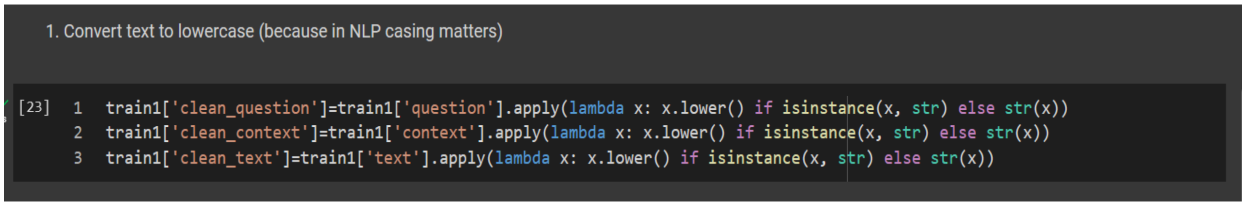


Checking for Null values

```
train1.isnull().sum()
#no null values
```

```
index               0
question            0
context             0
text            43498
answer_start    43498
c_id                0
dtype: int64
```

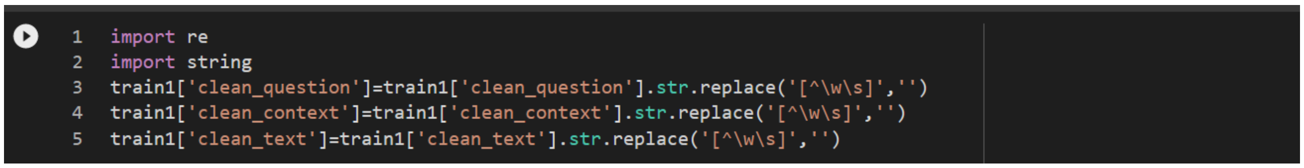
3) Convert text to lower-case



1. Convert text to lowercase (because in NLP casing matters)

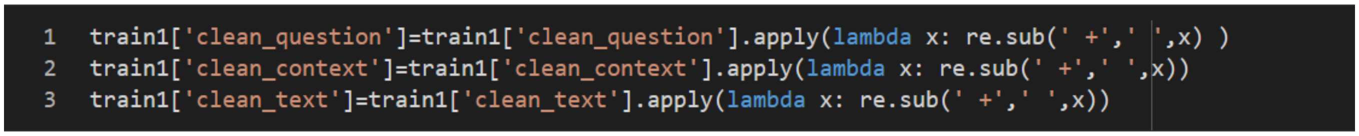
```
train1['clean_question']=train1['question'].apply(lambda x: x.lower() if isinstance(x, str) else str(x))
train1['clean_context']=train1['context'].apply(lambda x: x.lower() if isinstance(x, str) else str(x))
train1['clean_text']=train1['text'].apply(lambda x: x.lower() if isinstance(x, str) else str(x))
```

4) Remove punctuations



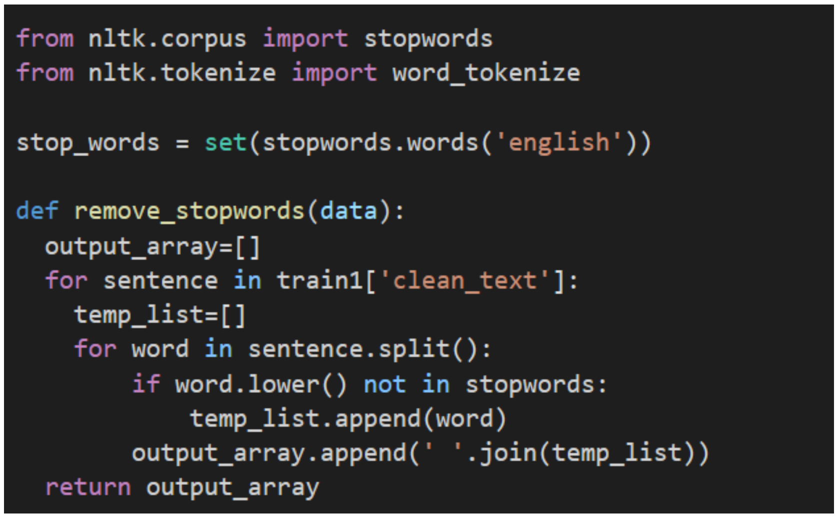
```
import re
import string
train1['clean_question']=train1['clean_question'].str.replace('[^\w\s]','')
train1['clean_context']=train1['clean_context'].str.replace('[^\w\s]','')
train1['clean_text']=train1['clean_text'].str.replace('[^\w\s]','')
```

5) Remove extra space



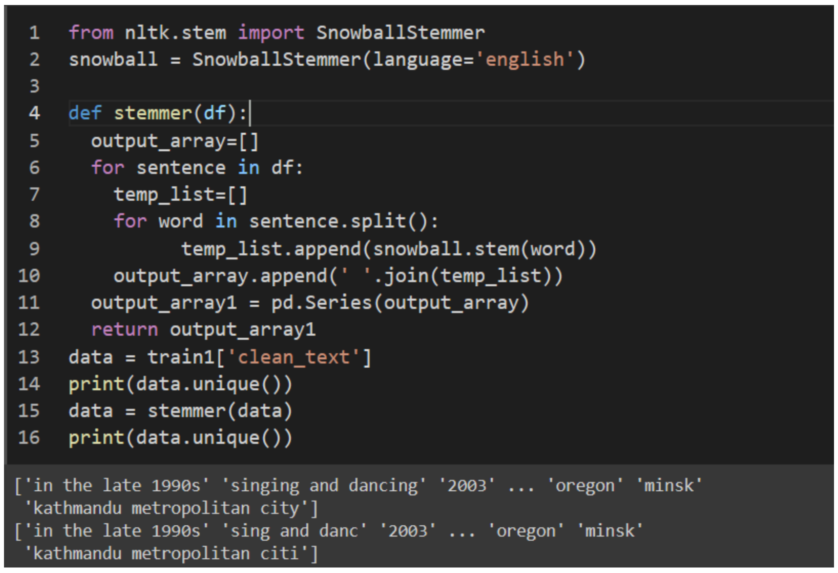
```
train1['clean_question']=train1['clean_question'].apply(lambda x: re.sub(' +',' ',x) )
train1['clean_context']=train1['clean_context'].apply(lambda x: re.sub(' +',' ',x))
train1['clean_text']=train1['clean_text'].apply(lambda x: re.sub(' +',' ',x))
```

6) Stop words removal



```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

stop_words = set(stopwords.words('english'))

def remove_stopwords(data):
  output_array=[]
  for sentence in train1['clean_text']:
    temp_list=[]
    for word in sentence.split():
        if word.lower() not in stopwords:
            temp_list.append(word)
        output_array.append(' '.join(temp_list))
  return output_array
```

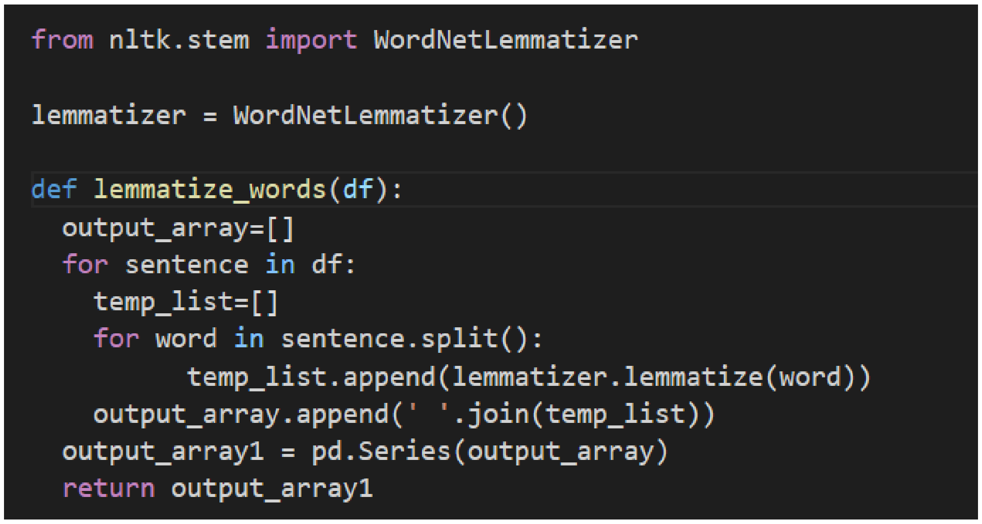
7) Stemming



```
from nltk.stem import SnowballStemmer
snowball = SnowballStemmer(language='english')

def stemmer(df):
  output_array=[]
  for sentence in df:
    temp_list=[]
    for word in sentence.split():
        temp_list.append(snowball.stem(word))
    output_array.append(' '.join(temp_list))
  output_array1 = pd.Series(output_array)
  return output_array1
data = train1['clean_text']
print(data.unique())
data = stemmer(data)
print(data.unique())
```

```
['in the late 1990s' 'singing and dancing' '2003' ... 'oregon' 'minsk'
 'kathmandu metropolitan city']
['in the late 1990s' 'sing and danc' '2003' ... 'oregon' 'minsk'
 'kathmandu metropolitan citi']
```

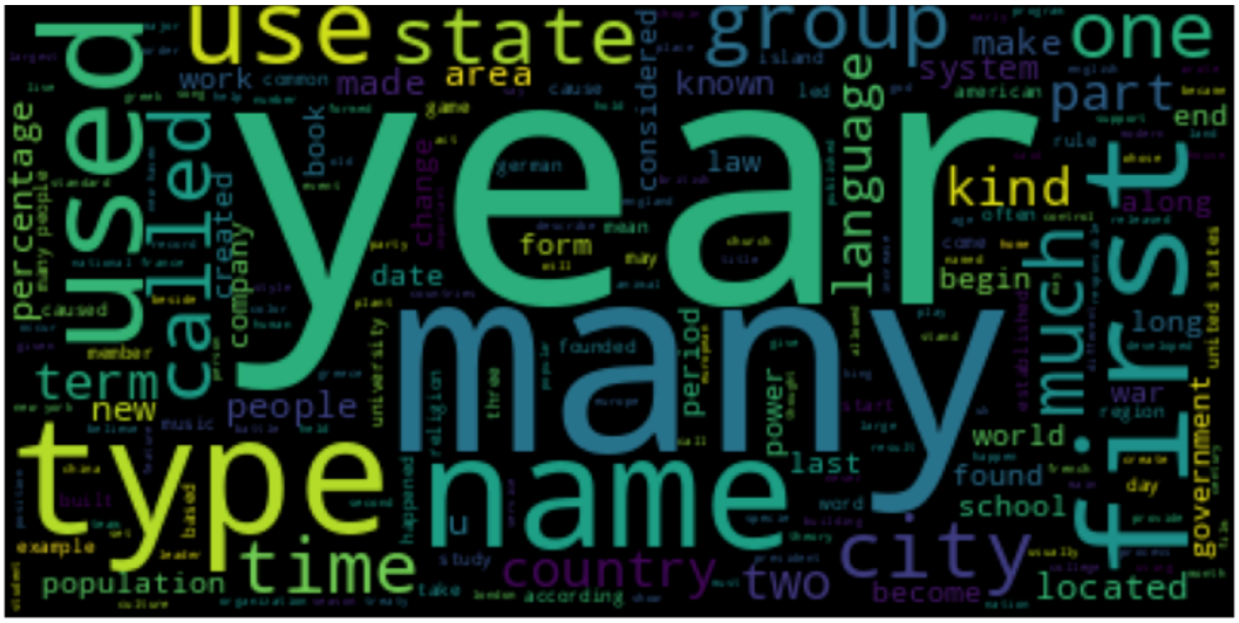
8) Lemmatization



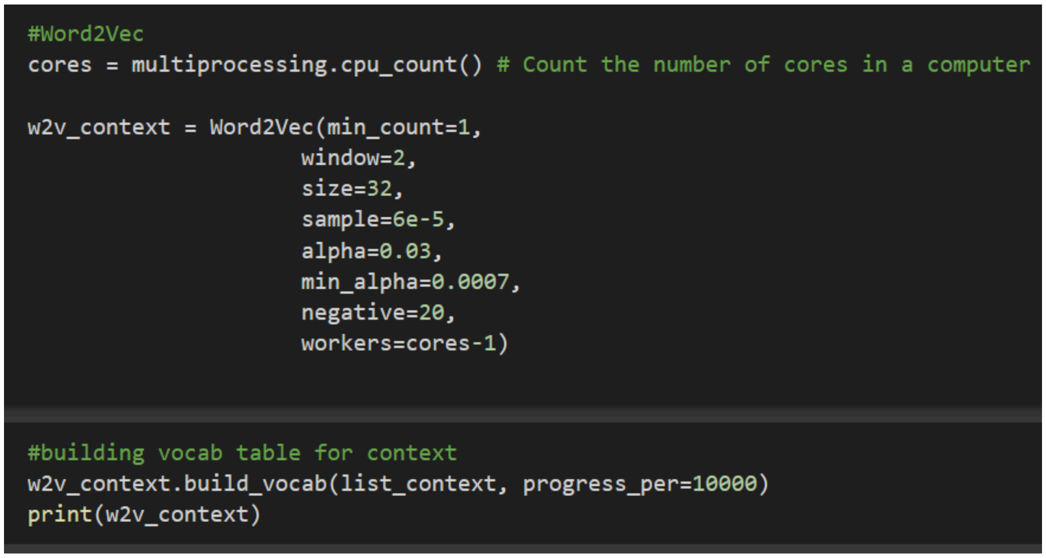
```python
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

def lemmatize_words(df):
  output_array=[]
  for sentence in df:
    temp_list=[]
    for word in sentence.split():
          temp_list.append(lemmatizer.lemmatize(word))
    output_array.append(' '.join(temp_list))
  output_array1 = pd.Series(output_array)
  return output_array1
```

<u>Exploratory Data Analysis and word embeddings</u>

1) Word- Cloud

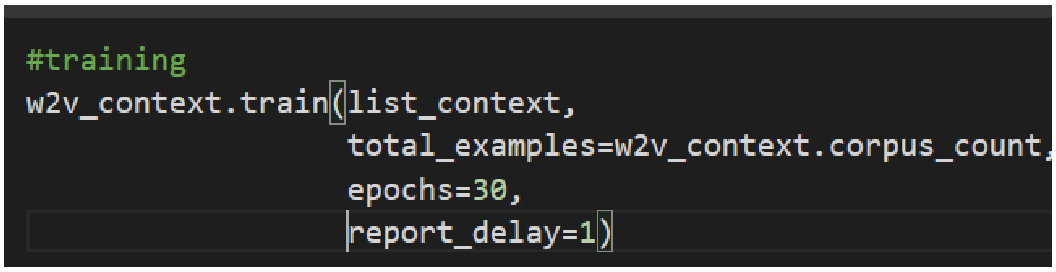


2) Word embeddings



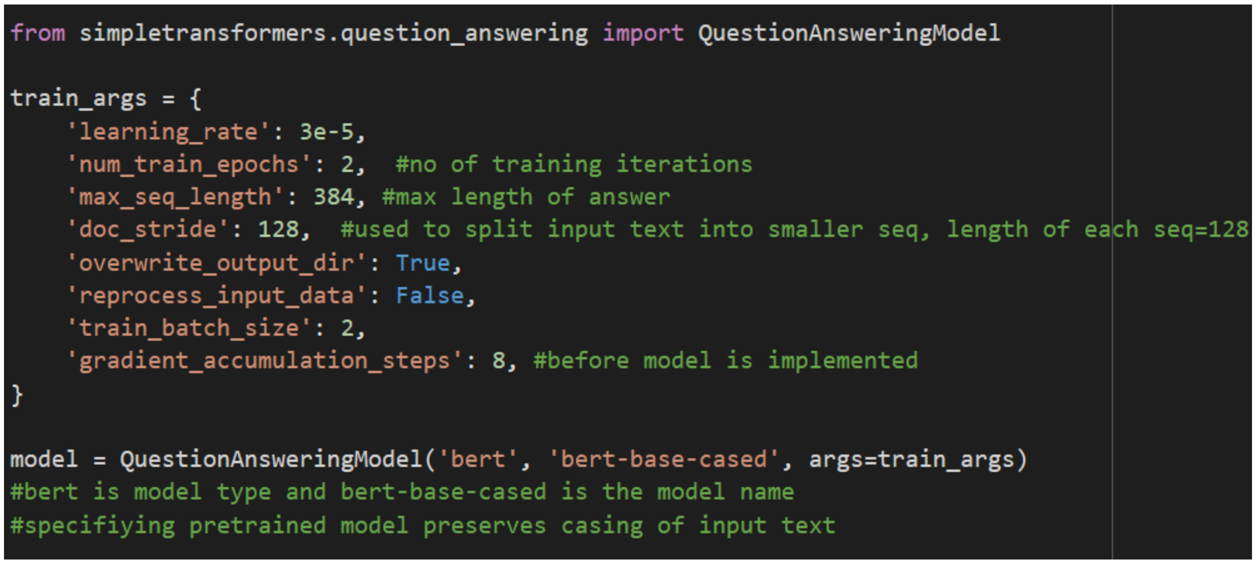
```
#Word2Vec
cores = multiprocessing.cpu_count() # Count the number of cores in a computer

w2v_context = Word2Vec(min_count=1,
                       window=2,
                       size=32,
                       sample=6e-5,
                       alpha=0.03,
                       min_alpha=0.0007,
                       negative=20,
                       workers=cores-1)
```

```
#building vocab table for context
w2v_context.build_vocab(list_context, progress_per=10000)
print(w2v_context)
```

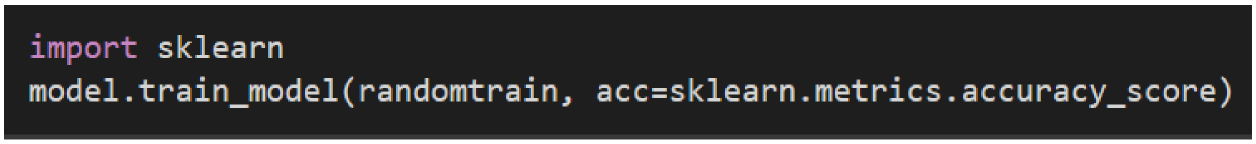


```
#training
w2v_context.train(list_context,
                  total_examples=w2v_context.corpus_count,
                  epochs=30,
                  report_delay=1)
```

Model Creation and training

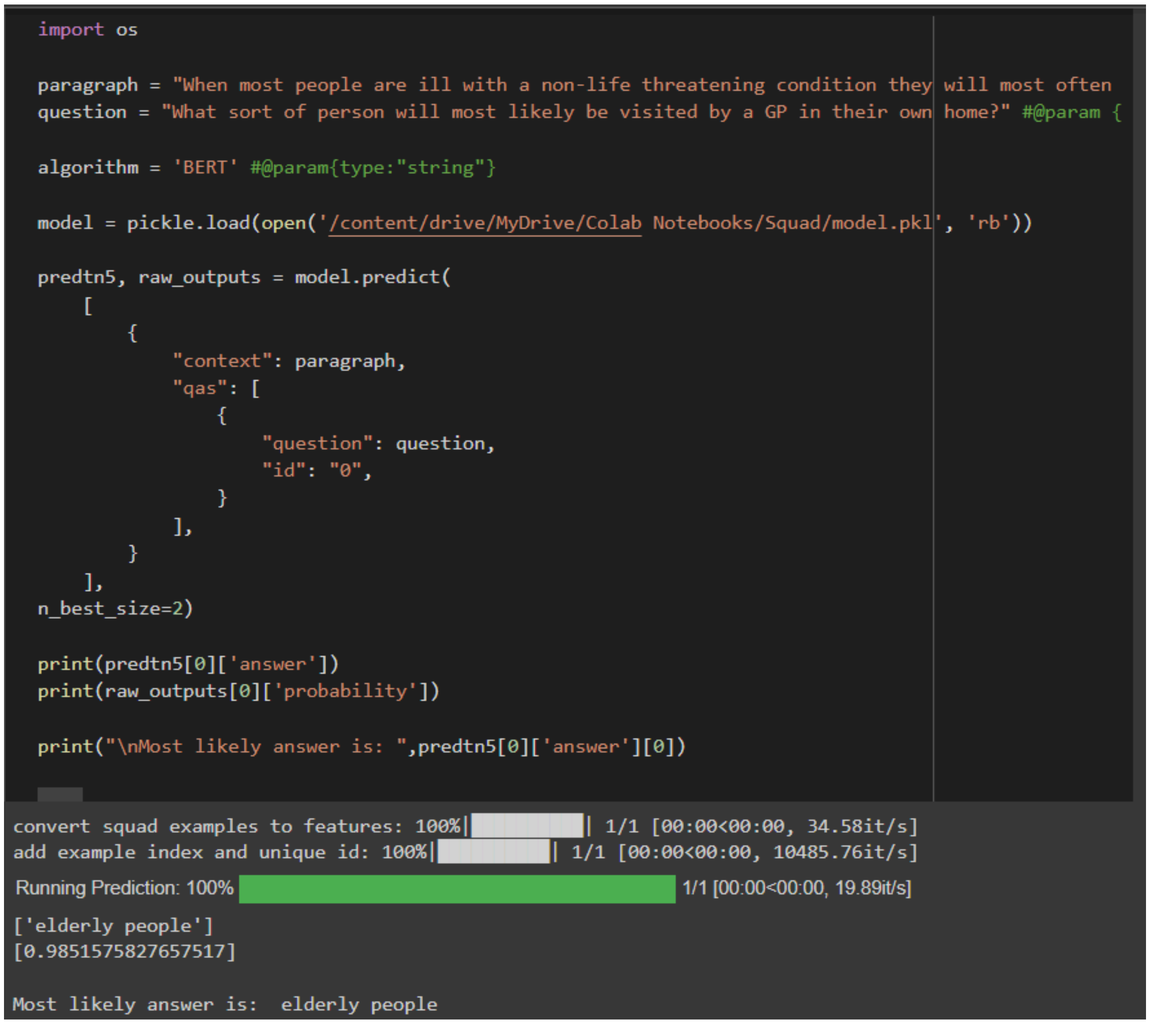


```
from simpletransformers.question_answering import QuestionAnsweringModel

train_args = {
    'learning_rate': 3e-5,
    'num_train_epochs': 2,  #no of training iterations
    'max_seq_length': 384, #max length of answer
    'doc_stride': 128,  #used to split input text into smaller seq, length of each seq=128
    'overwrite_output_dir': True,
    'reprocess_input_data': False,
    'train_batch_size': 2,
    'gradient_accumulation_steps': 8, #before model is implemented
}

model = QuestionAnsweringModel('bert', 'bert-base-cased', args=train_args)
#bert is model type and bert-base-cased is the model name
#specifiying pretrained model preserves casing of input text
```



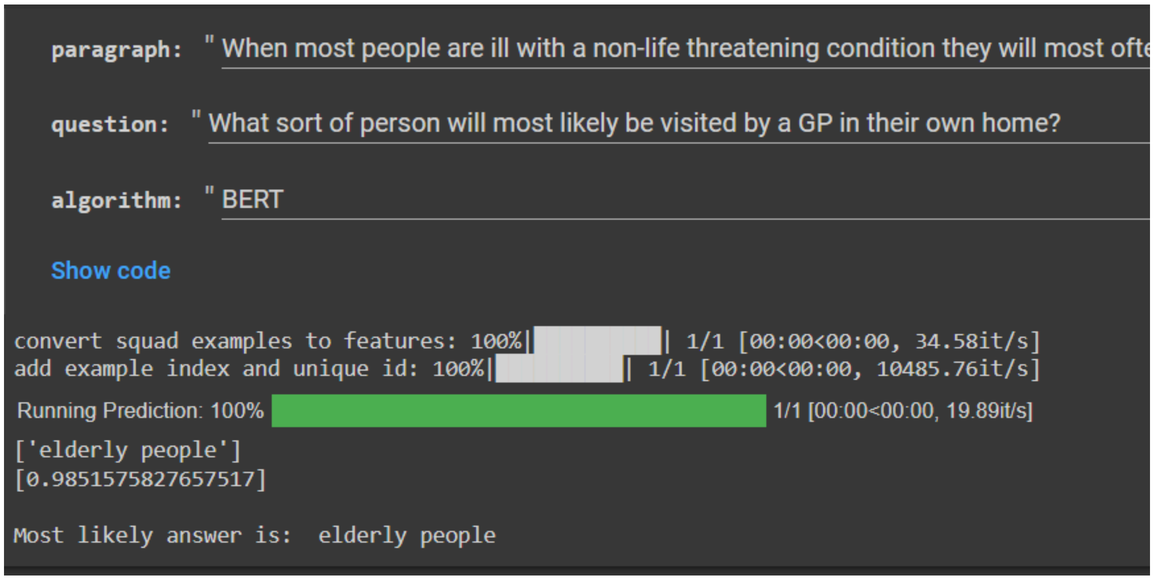
```
import sklearn
model.train_model(randomtrain, acc=sklearn.metrics.accuracy_score)
```

5.2 Sample Output screen



```
import os

paragraph = "When most people are ill with a non-life threatening condition they will most often
question = "What sort of person will most likely be visited by a GP in their own home?" #@param {

algorithm = 'BERT' #@param{type:"string"}

model = pickle.load(open('/content/drive/MyDrive/Colab Notebooks/Squad/model.pkl', 'rb'))

predtn5, raw_outputs = model.predict(
    [
        {
            "context": paragraph,
            "qas": [
                {
                    "question": question,
                    "id": "0",
                }
            ],
        }
    ],
n_best_size=2)

print(predtn5[0]['answer'])
print(raw_outputs[0]['probability'])

print("\nMost likely answer is: ",predtn5[0]['answer'][0])
```

```
convert squad examples to features: 100%|██████████| 1/1 [00:00<00:00, 34.58it/s]
add example index and unique id: 100%|██████████| 1/1 [00:00<00:00, 10485.76it/s]
Running Prediction: 100%  1/1 [00:00<00:00, 19.89it/s]
['elderly people']
[0.98515758276575 17]

Most likely answer is:  elderly people
```



paragraph: " When most people are ill with a non-life threatening condition they will most ofte

question: " What sort of person will most likely be visited by a GP in their own home?

algorithm: " BERT

Show code

```
convert squad examples to features: 100%|██████████| 1/1 [00:00<00:00, 34.58it/s]
add example index and unique id: 100%|██████████| 1/1 [00:00<00:00, 10485.76it/s]
Running Prediction: 100%  1/1 [00:00<00:00, 19.89it/s]
['elderly people']
[0.9851575827657517]

Most likely answer is:  elderly people
```

6. Conclusion and Future Enhancement:

In conclusion, the BERT-based passage-based question answering model represents a significant development in machine learning and natural language processing. BERT is a useful tool in the field of question-answering systems due to its capacity to comprehend the context of words in a phrase and effectiveness in finding semantic linkages between them.

The passage-based technique has significantly expanded BERT's capabilities because it allows it to evaluate full paragraphs or passages of text rather than just single phrases. This means that it can provide more accurate and comprehensive answers, making it an important tool for tasks such as information retrieval, document classification, and sentiment analysis as it can offer more accurate and thorough responses to challenging inquiries.

The passage-based question-answering paradigm with BERT has demonstrated considerable promise in the area of natural language processing and has the ability to change how we interact with text-based data. We may anticipate ever more sophisticated models and methodologies to develop as technology advances, which will move us closer to the objective of building really intelligent machines that can comprehend human language.

There are several possible future enhancements for a closed-domain passage based question answering system using NLP and transfer learning (BERT):

One enhancement is to fine-tune the BERT model on specific domains to improve the accuracy of the system. This can be done by fine-tuning the model on a large corpus of domain-specific texts, such as legal or medical documents, and then retraining the model on the specific question answering task.
Another enhancement is to use multi-task learning to train the model to perform multiple related tasks simultaneously. For example, the model can be trained to answer questions, summarize passages, and extract named entities from text, all at the same time.
To improve the accuracy of the system, external knowledge sources such as knowledge graphs, ontologies, and dictionaries can be integrated into the model. This can help the system to answer more complex questions that require external knowledge.

7. Reference:

[1] Yang, W., Xie, Y., Lin, A., Li, X., Tan, L., Xiong, K., ... & Lin, J. (2019). End-to-end open-domain question answering with bertserini. arXiv preprint arXiv:1902.01718.

[2] Wang, Z., Ng, P., Ma, X., Nallapati, R., & Xiang, B. (2019). Multi-passage bert: A globally normalized bert model for open-domain question answering. arXiv preprint arXiv:1908.08167.

[3] Pan, L., Chakravarti, R., Ferritto, A., Glass, M., Gliozzo, A., Roukos, S., ... & Sil, A. (2019). Frustratingly easy natural question answering. arXiv preprint arXiv:1909.05286.

[4] Caciularu, A., Dagan, I., Goldberger, J., & Cohan, A. (2022, July). Long Context Question Answering via Supervised Contrastive Learning. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 2872-2879).

[5]Choi, E., He, H., Iyyer, M., Yatskar, M., Yih, W. T., Choi, Y., ... & Zettlemoyer, L. (2018). QuAC: Question answering in context. arXiv preprint arXiv:1808.07036.

[6] Alberti, C., Lee, K., & Collins, M. (2019). A bert baseline for the natural questions. arXiv preprint arXiv:1901.08634.

[7] Glass, M., Gliozzo, A., Chakravarti, R., Ferritto, A., Pan, L., Bhargav, G. P., ... & Sil, A. (2019). Span selection pre-training for question answering. arXiv preprint arXiv:1909.04120.

[8] Zaib, M., Tran, D. H., Sagar, S., Mahmood, A., Zhang, W. E., & Sheng, Q. Z. (2021). BERT-CoQAC: BERT-based conversational question answering in context. In Parallel Architectures, Algorithms and Programming: 11th International Symposium, PAAP 2020, Shenzhen, China, December 28–30, 2020, Proceedings 11 (pp. 47-57). Springer Singapore.

[9] Izacard, G., & Grave, E. (2020). Leveraging passage retrieval with generative models for open domain question answering. arXiv preprint arXiv:2007.01282.

[10] Guo, M., Yang, Y., Cer, D., Shen, Q., & Constant, N. (2020). Multireqa: A cross-domain evaluation for retrieval question answering models. arXiv preprint arXiv:2005.02507.

[11] Ahmad, A., Constant, N., Yang, Y., & Cer, D. (2019). ReQA: An evaluation for end-to-end answer retrieval models. arXiv preprint arXiv:1907.04780.