**Answering Questions using Multi-Head Attention on AmbigQA**

**Abstract**

Answering ambiguous open-domain questions is not an easy task. It can be challenging to answer those ambiguous questioning that have a individual or double answers. We are introducing Ambiguous Question Answering, a advanced open-domain questioning-analogous undertaking that finds all probable response and resolves the ambiguity. The dataset consists of 14,042 inquiry from the Natural Questions accessible-realm NQ-Open QA bench mark. We discover that more than one-part of the NQ-open queries abide unclear. NQ-OPEN has a variety of ambiguous sources, including references to events and entities. Our new challenge and results clearly show that our robust guideline models for AMBIGQA, whatever we showed interest from decrepit managed learning that incorporates NQ-OPEN, will enable a significant future research effort. By using multi-head attention in the Transformer model, we can find clear answers that may be in single or multiple answers, and those questions and answers must and should be in the dataset. The decoder and encoder sides of the Transformer model, which uses self-attention, The transformer model gives the best result for Questioning Answering task and Recall-Oriented Understudy for Gisting Evaluation (Rouge Score). The evaluation metric for the Rouge score is very good.

**Keywords:** Answering, AMBIGQA, Natural language Processing, Transformer Model.

**Introduction**

The ambition of a question-answering scheme is to create programmes that can respond to queries in natural language that are posed by users. A Enquiry responsive system is typically developed to excerpt solution from a database that is structured or from a collection of unstructured natural language document(s). Answering questions comes in two different forms. One is a closed-domain question-answering system, which primarily handles inquiries inside a single domain, and the other is open-sphere question-respondent scheme, which handles inquiries connected almost anything. Here we are using open-domain question respondent system on AmbigQA. The questioning and answering tasks are too difficult to develop clear and unambiguous queries in an open-domain environment. As stated by Voorhees et al. (1999), conventional approaches presumptively presume that there is only one clear answer to a question. Open-domain QA systems are intended to respond to any type of question that gives relevant answers to those questions, and those Question Answering should be in a dataset. When questions are written while information is being gathered (such as search terms), ambiguity frequently occurs in open-domain question answering. To investigate this problem,AMBIGQA (Responding Ambiguous Open-domain Queries) is a new activity that entails disambiguating and responsive possibly indeterminate questions. The model must

1. discover a fit of unique, unevenly reasonable solutions to the question and

(2) supply simple so far absolute compose of the questions that modify the explanation that leads to each response. Additionally, the NQOPEN open-domain version of NATURAL QUESTIONS, which has 14,042 questions, is used in the AMBIGNQ dataset (Kwiatkowski et al., 2019).

## Problem Definition

## Previous question answering system models were unable to provide faultless responses in comparison to our approach. Consider the preceding model: it gives graph-based solutions, BERT and BART, Mimics logs, and so on. Our Transformer model provides the best outcomes to the question answering system on AMBIGQA, which can provide a single or numerous replies.

## Motivation & Project Scope

A specific type of information retrieval called question answering looks for knowledge. Not only are we interested in finding the pertinent sites, but we also want to get answers to our questions that are precise. “Human Language Technology, IR, ML, Knowledge Discovery, Philosophy, and Assumption, Search all cross-disciplinary with query-answering”[1]. A question-and-answer system is inherently an artistic and scientific effort. Everywhere, including medical science, student learning systems, and personal assistants, there is a need for question-answering systems. Every area of our lives where we require technological aid is a requirement. Of course it makes sense to look into the exciting topic of question answering.

Answering Questions with Multi-Head Attention on AMBIGQA is a difficult job in the tract of NLP, and assorted approaches, such as linguistic-based approaches, statistical-based approaches, pattern matching approaches, and others, have been proposed. The goal of this project is to recognize and predict ambiguous questions. So as to detect the correct answers , By employing word segmentation, part-of-speech tagging, and word lemmatization to annotate fundamental linguistic features in the input, we will use it as a query. The resulting data is compared to the document on the databases in order to predict or recognize the correct answers, which may be in a single line or multiple lines.

## Proposed System

## Here we will suggest a summary of the tool for answering questions using Multi-Head Attention on AMBIGQA.

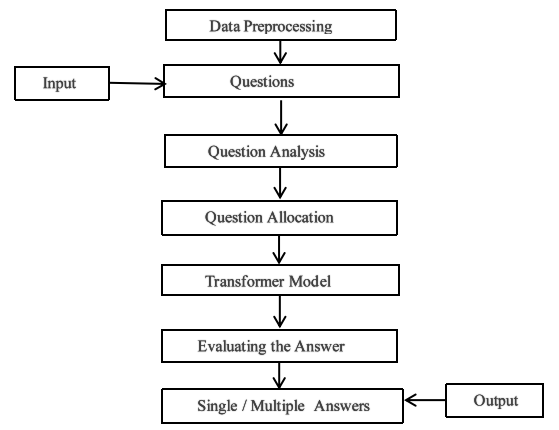


Figure :- Flow-chart

Data preprocessing is a data mining approach that involves converting raw data into a usable format. The dataset consists of 10036 rows and 7 columns. The dataset consists of Viewed\_doc\_titles, Used\_queries, Annotations, NQ\_answer, ID, NQ\_doc\_title, and Question. Once the data preprocessing is done, the input is given as questions, and then Question Analysis is performed, which entails examining the questions to extract the required information for establishing what is being asked and how to approach answering it. In a dataset, there are certain topics related to those questions, and it examines them. After performing the question analysis, the question allocation is done; it is an important component in inquiry answering systems since it assists in determining the kind of questioning and its matching form of solution.

After operating the question allocation task by using the transformer model, it can detect the unambiguous answer. The Transformer model intends to handle long-range relationships with ease while resolving sequence-to-sequence operations. It does not use sequence-aligned RNNs or convolution to correspond of its input and output; rather, it completely relies on self-attention.The transformer model has encoded and decoded layers; those have the same amount of entities (N = 6). The transformer model is used in this case for examining the answer based on the question asked. It provides either single or multiple answers.

for example

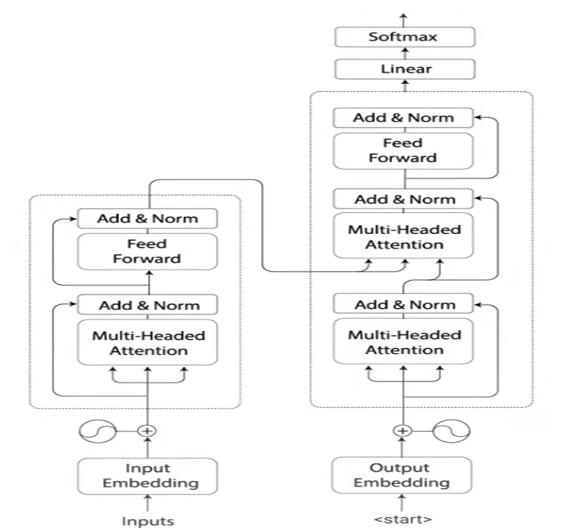
1. When did Epcot's Frozen Ride debut? Single answer: June 21, 2016
2. What is the legal age of marriage in the USA? Multiple answers: 19 years, 21 years, 23 years, and soon…

As seen in the example, output is the answer to a single or multiple answers.

**Transformer Model**

The Transformer is the introductory transduction model to generate depiction of its input signal and output signal exclusively utilising self-attention, rather than convolution or sequence-aligned RNNs. Transforming input sequences into output sequences is referred to as "transduction" in this context.

The goal of Transformer is to handle input and output relationships with complete attention and recurrence. A transformer model is a type of encoder-decoder network that employs self-attention on both the encoder and decoder sides as shown in the figure 3.2.



Positional Encoding

Positional Encoding

Figure 2:- Transformer Model

### **Data Set**

[Evaluation script](https://github.com/shmsw25/AmbigQA" \t "https://nlp.cs.washington.edu/ambigqa/_blank) / [Eval script README](https://github.com/shmsw25/AmbigQA/" \l "evaluation-script" \t "https://nlp.cs.washington.edu/ambigqa/_blank) / [Data content README](https://github.com/shmsw25/AmbigQA/" \l "dataset-contents" \t "https://nlp.cs.washington.edu/ambigqa/_blank)

[Download AmbigNQ (light ver.)](https://nlp.cs.washington.edu/ambigqa/data/ambignq_light.zip)

[train/dev] question & answers (1.1M)

[Download AmbigNQ (full ver.)](https://nlp.cs.washington.edu/ambigqa/data/ambignq.zip)

[train/dev] question, answers, original NQ answer, visited Wikipedia pages, used search queries & search results (18M)

[Download AmbigNQ (evidence ver.)](https://nlp.cs.washington.edu/ambigqa/data/ambignq_with_evidence_articles.zip)

[train/dev/test] question, answers (except test), semi-oracle evidence articles ([details](https://github.com/shmsw25/AmbigQA/blob/master/evidence.md" \t "https://nlp.cs.washington.edu/ambigqa/_blank)) (575M)

[Download NQ-open](https://nlp.cs.washington.edu/ambigqa/data/nqopen.zip)

[train/dev/test] question, NQ answer & associated document (3.9M)

### Additional Resources

* Wikipedia DB from 01-20-2020 in sqlite db, consistent to [DrQA](https://github.com/facebookresearch/DrQA/tree/master/scripts/retriever" \t "https://nlp.cs.washington.edu/ambigqa/_blank) [[plain text (5.0GB)](https://nlp.cs.washington.edu/ambigqa/data/docs.db.zip)] [[html preserved (7.7GB)](https://nlp.cs.washington.edu/ambigqa/data/docs-html.db.zip)]
* Wikipedia DB from 01-20-2020 in .tsv.gz, consistent to [DPR](https://github.com/facebookresearch/DPR" \t "https://nlp.cs.washington.edu/ambigqa/_blank) [[.tsv.gz (4.8GB)](https://nlp.cs.washington.edu/ambigqa/data/psgs_w100_20200201.tsv.gz)]
* ****Update 07/2020:**** [Baseline codes (DPR and SpanSeqGen)](https://github.com/shmsw25/AmbigQA/tree/master/codes" \t "https://nlp.cs.washington.edu/ambigqa/_blank)are available now, along with [model checkpoints](https://github.com/shmsw25/AmbigQA/tree/master/codes" \l "need-preprocessed-data--pretrained-models--predictions" \t "https://nlp.cs.washington.edu/ambigqa/_blank).

## **Technologies/Languages Used**

**Pandas**

An open source Python package called Pandas offers a framework for creating tabular data and performing row, column, and cell transformations. It is a helpful tool for gathering, analysing, and mining data for knowledge as well as structuring it in a way that machine learning algorithms can use it.

**Numpy**

The pandas library and the free source NumPy library are used to manage multidimensional data and carry out intricate scientific and mathematical operations on the data. Numerous mathematical operations, including trigonometric, statistical, and algebraic routines, are performed on arrays using this method.

**Matplotlib**

As part of NumPy, a tool for handling big data numerically, Matplotlib is a plotting library for the Python programming language. To integrate charts into Python programmes, Matplotlib makes use of an object-oriented API. A multi-platform data visualisation package called Matplotlib is based on numpy arrays and made to interact with the entire scipy stack.

**Torch**

Torch is an open-source unscientific computation framework, scripting language, and machine learning library built on the Lua scheduling language. It leverages the scripting language LuaJIT and an underlying C code to deliver a wide variety of deep learning methods. It offers a versatile N-dimensional array called a Tensor that supports the fundamental operations of indicator, wedge, trans versing, type-flinging, resuming, sharing stockpile, and replicate. Since the majority of other packages depend on this object, it serves as the library's main object. The Tensor additionally supports BLAS use like dot commodity, matrix-vector multiplication, matrix-matrix multiplication, and matrix product in addition to scientific operations like max, min, and sum as well as numerical dissemination like unvarying, normal, and multiracial. A single number will be used to generate the tensor after the import of the torch module ( Mean and Standard Deviation). Both tensors must contain the same amount of elements.

**Torch.Optim**

It is a package that executes different optimization techniques. The interface is sufficiently generic and already supports the majority of widely used techniques.

**Torch Text**

It consists of tools for data processing and well-known natural language datasets.

#import torchtext from torchtext.data import Field, BucketIterator, TabularDataset

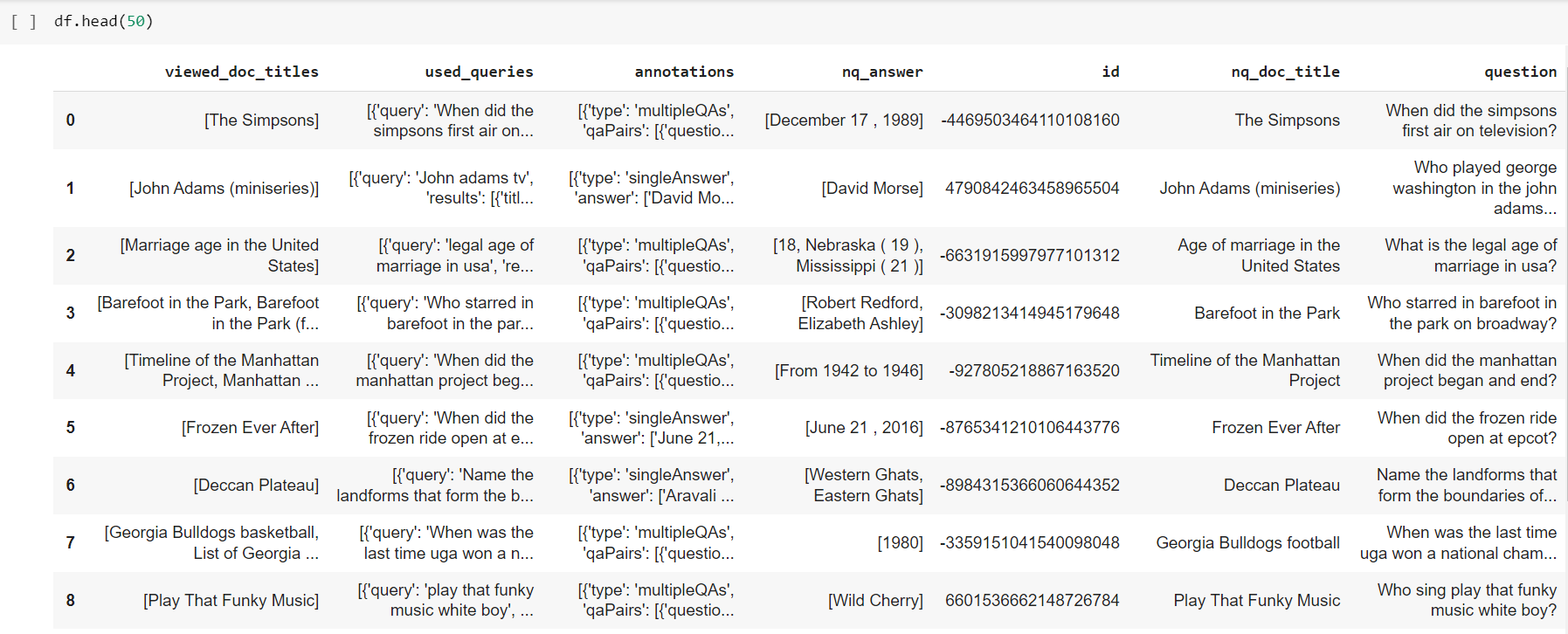
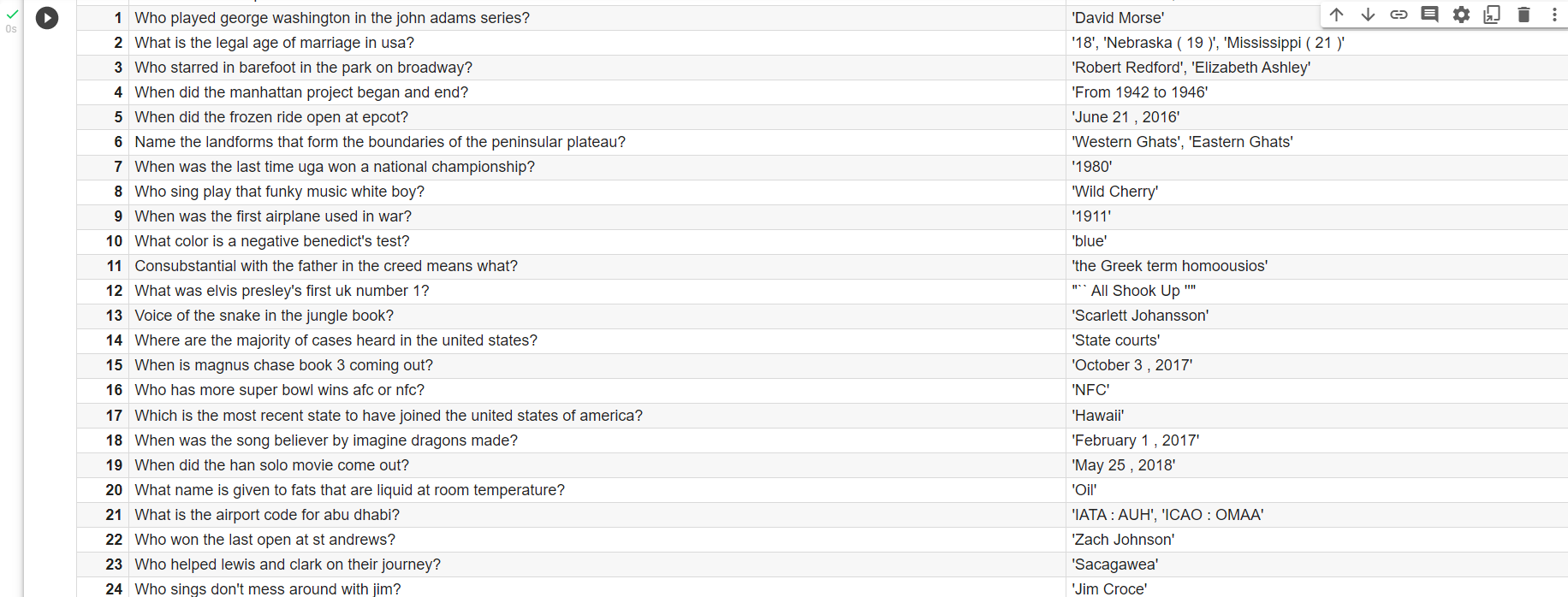
en =**spacy.load ('en') fr = spacy.load ('fr') def tokenize\_en (sentence):**

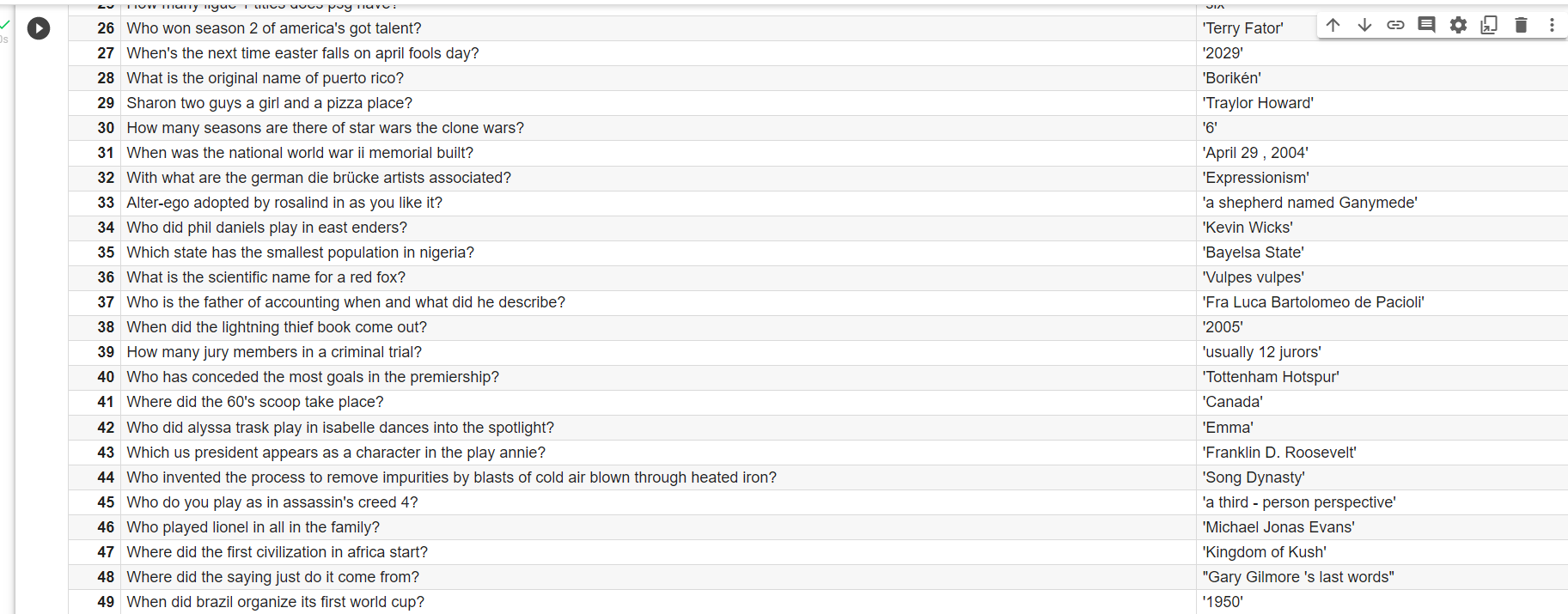
**Torch Text.Legacy**

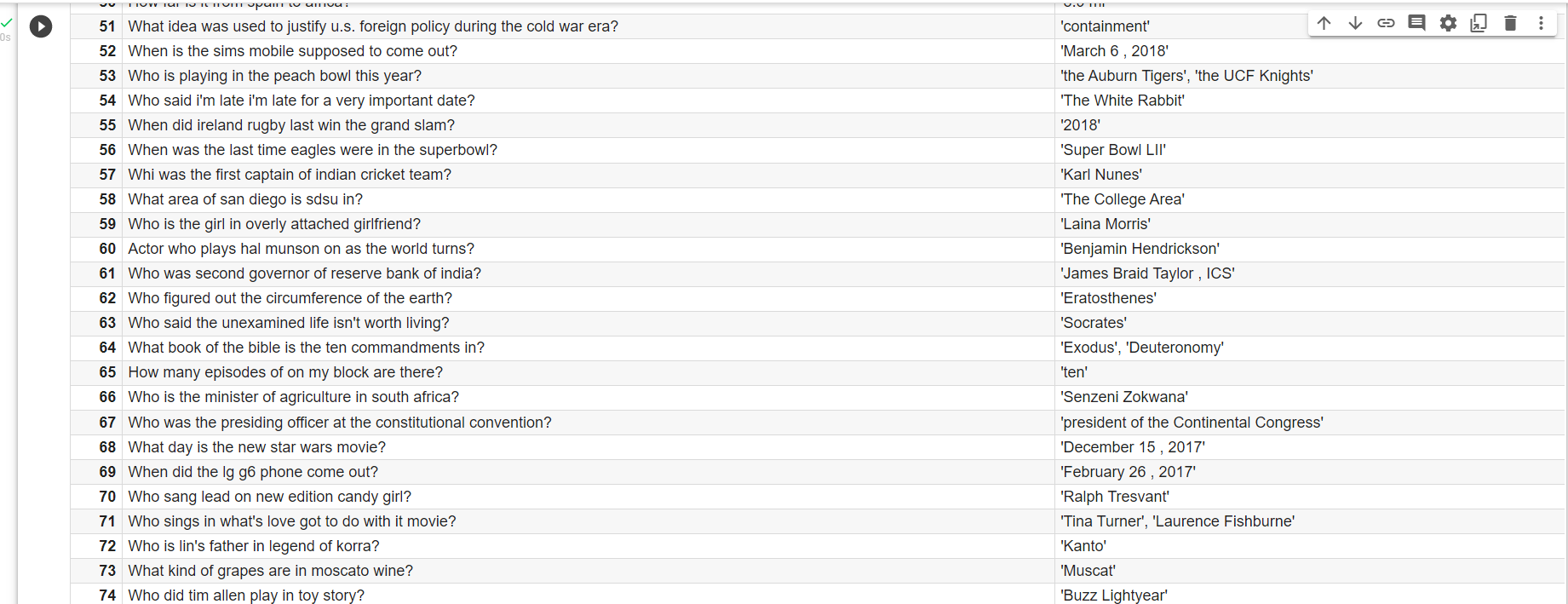
In the most recent version of Torch Text, several functionalities were transferred to Torch Text.legacy.

**Output**

It produces seven columns and 10036 rows, which have doc\_titles, queries, annotations, answers, IDs, doc\_titles, questions.

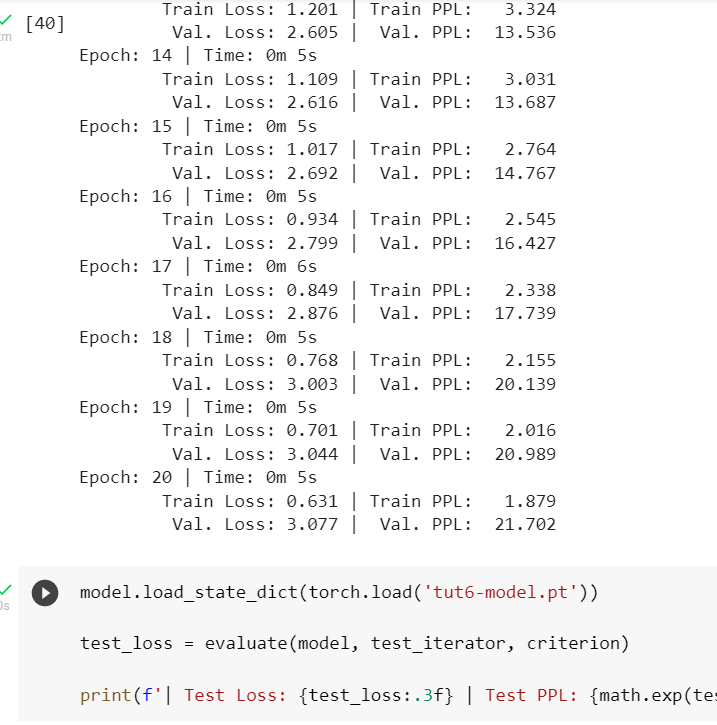
It is the output of the Answering questions using multi-head attention on AMBIGQA. It provides single or multiple answers based on the questions asked in the data set.

****

****



Epoch’s informs us about the loss and perplexity of a train and its value.



The evaluation metric (ROUGE Score) is depicted in the graphic below.

