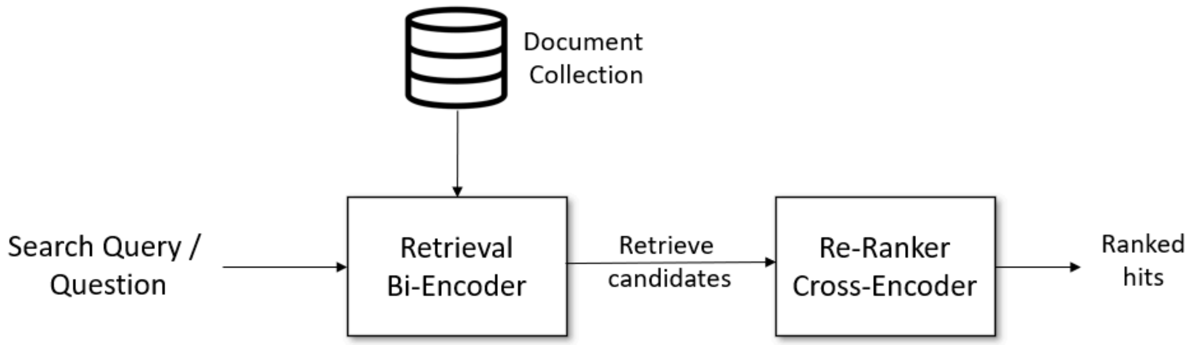
Team name in CodaLAB : Atul, abhishek and et al

Retriever

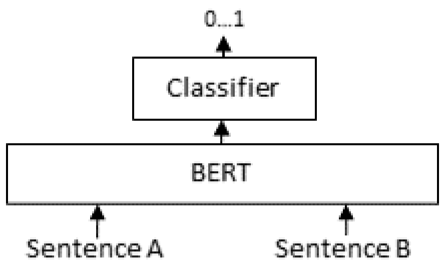
Logic:

We utilized SentenceTransformer to generate embeddings for lines of the Wikipedia data. Initially, a retrieval system retrieves a sizable list, such as 100, of potentially relevant hits for a given search query. Dense retrieval with a cross-encoder is often employed for this purpose.



However, the retrieval system may sometimes retrieve documents that are not closely related to the search query. Hence, in a subsequent stage, we employ a re-ranker based on a cross-encoder to assess the relevance of all candidates for the provided search query.

The retrieval process must be efficient, especially for large document collections with millions of entries. Nonetheless, it may occasionally produce irrelevant candidates. A cross-encoder-based re-ranker significantly enhances the final outcomes for the user. This model simultaneously evaluates the query and a potential document, generating a single relevance score between 0 and 1.



Source: https://www.sbert.net/examples/applications/retrieve_rerank/README.html

Additionally, we performed Named Entity Recognition (NER) on the provided sentences. If at least 50% of the named entities are not present in the retrieved sentences, we discard those retrieved sentences. This step aids the model in better classifying the claim into the "NOT ENOUGH INFO" category.

Results:

Recall@1:0.19
Recall@10:0.3477
Recall@100: 0.707

Natural Language inference:

Logic:

We utilized the T5-small model, a state-of-the-art transformer-based architecture, to analyze and categorize claims into three classes: SUPPORTED, REFUTED, or NOT ENOUGH INFO. To achieve this, we leveraged the power of transfer learning by utilizing pre-trained embeddings from the T5 model's last layer. These embeddings capture intricate semantic features of the input text.

Next, we extended the model by adding a classification layer on top of these embeddings. This additional layer, implemented as a linear transformation (nn.Linear), maps the T5 embeddings to probabilities across the three output classes.

During the fine-tuning process, we aimed to enhance the model's performance on our specific task. To achieve this, we utilized a technique known as transfer learning. We froze the parameters of the T5 model's last layer to preserve the pre-learned knowledge, preventing it from being updated during training. Then, we fine-tuned the entire model, including the classification layer, using the provided training data from the train.py file. This process allows the model to adapt its parameters to better fit the nuances and patterns present in the training data, ultimately improving its ability to classify claims accurately.

Results:

Precision: 93% (if gold standard retriever is used)