Project Writeup

By Zihan Ma, Jake Baldwin, Abhay Patel, Seyed Ghaemi

What we did?

Basic model

First we used the sentenceBERT model ('all-mpnet-base-v2') to generate vector embeddings for entire sentences.

Originally, we finetuned the BERT model by picking question pairs at random, then assigning scores to how well the sentences match using some basic features.

For the basic finetuning features:

- Every question pair started with a score of 0.1
- We added 0.2 if they had the same category
- We added 0.6 if they had the same answer

Once we have these embeddings, we use the same process as the TF-IDF model from class except we're using our new BERT embeddings. We compare the cosine similarity to find our predicted answer. Lastly, for feature engineering, we use:

- Has our answer changed from the previous part of the question
- The percent of the top guess's score compared to sum of all guesses
- Probability of of category given the guess

Developing better features

The main change we planned to make was changing the finetuning features we were using to utilize named entity linking. In order to obtain named entities in each question we used the nltk library's NER. To link named entities together, we used DELFT's Wikipedia knowledge graph where each Wikipedia article was a node in the graph that contained other Wikipedia articles that were related to it with a score for each connected article. We used these nodes in our finetuning as follows:

1. For every Wikipedia title, we stored the top 20 linked entities
2. For each sentence, we generated named entities using nltk
   - For each named entity, we found its linked entities using (1) and combined them all together
     - The linked entity list should follow the format: `[[Entity_1, Score_1],
       [Entity_2, Score_2], ...]`
   - If the entity did not exist in the graph, we put that in a separate list of missing named entities instead

- ○ If the same entity appeared multiple times, we merged them into one entity and took the mean of their scores
3. After picking 2 questions randomly for finetuning:
    - ○ Initialized `count = 0` and for each matching entity across the 2 questions, we did `count += 2 * (1 - (score_1 - score_2) ** 2)` where `score_1` is from question 1's matched entity, and `score_2` is from question 2's matched entity.
    - ○ Checked how many of the missing entities of each question matched together, we called this j

4.
$$\text{final\_score} = \frac{2j}{p_1 + p_2} \frac{p_1 + p_2}{(p_1 + p_2)(k_1 + k_2)} + \frac{\text{count}}{s_1 + s_2} \frac{k_1 + k_2}{(p_1 + p_2)(k_1 + k_2)}$$

- ○ k_i is the length of the linked entity list of question i
- ○ p_i is the length of the list of entities without wikipedia nodes for question i
- ○ s_i is the length of all entities in question i

After finetuning, the rest of the steps were the same as before.

Why was it a good idea?

Many of the downfalls in all our prior systems were related to the fact that the system could not establish relationships between terms. On top of that, it was looking for keywords, rather than named entities or things that are wikipedia page articles. Using linked named entities to encode our sentences, should give the model a better understanding of relationships outside of what is simply in the question.

Using sentenceBERT rather than Word2Vec to create encodings was another good idea for several reasons. For one BERT learns to maintain critical and ignore useless long distance dependencies across words using its transformers. In addition, BERT works like a bidirectional LSTM which means it can look forward and backwards in order to understand what a particular word means.

Did our technique work or not?

Who did what?

- sentenceBERT
    - ○ Zihan Ma, Jake Baldwin
- Named Entity Recognition
    - ○ Abhay Patel, Seyed Mohammad Ghaemi
- Knowledge Graph
    - ○ Abhay Patel, Seyed Mohammad Ghaemi
- Combining NER and Knowledge Graph into sentenceBERT

- Zihan Ma
- Packaging model into the QuizBowlModel
  - Jake Baldwin
- Writeup
  - Zihan Ma, Jake Baldwin, Abhay Patel, Seyed Ghaemi
- Presentation
  - Zihan Ma, Jake Baldwin, Abhay Patel, Seyed Ghaemi