Hiva Mohammadzadeh

CS288 HW5 Report

In the report, describe the algorithm that you implemented and intuition of why it may it could be effective.

For my converge based selection, I first initialized a pre-trained language model for sentence embeddings using the SentenceTransformer function from the sentence)transformers package. The function then computes embeddings for all the training examples by encoding them to the string: "Please parse the following sentence: [query]. Answer: [program]". The embeddings are obtained by encoding this concatenated string using the pre-trained language model. The utterance is also encoded using the pre-trained language model to obtain its embedding. I then, compute the cosine similarity between the utterance embedding and the embeddings of all the training examples using the util.cos_sim function from the sentence_transformers package. Then I get the example with the highest similarity as the first example in the prompt. The remaining examples are selected based on their diversity score. For each remaining example, the diversity score is computed as the average cosine similarity between the example and all previously selected examples. If an example has already been selected, its diversity score is set to 0. The example with the highest diversity score is selected, and its index is added to the list of selected indices. The selected examples are then used to construct the final prompt by concatenating the query and program of each selected example into a string. The utterance is added to the end of the string with the label "Answer:". The prompt is returned as a list of dictionaries.

My architecture is effective because it leverages the power of pre-trained language models to compute embeddings for training examples, which captures their semantic similarity. The use of cosine similarity to rank examples allows for a simple and efficient way to identify the most similar examples to the utterance. The addition of diversity scoring ensures that the prompt is constructed using a variety of examples, which can improve the robustness of the system and avoid overfitting to specific examples.

For the report, compare the predictions from the example selection methods using 1) uniform random sampling 2) embedding-based similarity search and 3) coverage based selection. Compare and contrast the errors and submit your analysis as report.pdf

My prediction accuracies were:

1. Uniform Random SamplingUniform sampling prompt: 0.275
2. Embedding-based similarity search - Nearest neighbor prompt: 0.425
3. Coverage based selection - Diversity based prompt: 0.5

Which show that the embedding-based similarity search and coverage-based selection methods are more effective than the uniform random sampling method for generating predictions that exactly match the given prompts. Both the nearest neighbor and diversity-based selection methods achieved an exact match rate of 0.425 and 0.5, indicating that they are more effective for this task than the random sampling. These results suggest that incorporating similarity and diversity criteria in example selection can improve the accuracy of predictions.

The uniform random sampling method had the highest error rate, suggesting that randomly selecting examples is not an effective strategy for this task. This model involves selecting examples from the training set randomly without considering any specific criteria. It may result in selecting examples that are not relevant or representative of the task, which may negatively impact model performance.

The embedding-based similarity search and coverage-based selection methods had lower error rates, but still made errors. This is because embedding-based similarity search involves constructing embeddings for each example in the training set and then finding examples that are most similar to the given utterance based on their embeddings, but it requires additional computation to construct embeddings and find the most similar examples. Coverage-based selection methods also involves selecting examples that maximize the coverage of the training set based on some criterion, but it may require additional knowledge about the task or the training set, and may be difficult to implement without such information.

Upon examining the errors made by the model, it appears that many of them were due to the model's inability to handle complex or unusual queries. For example, the model struggled with queries that contained negation, ambiguity, or uncommon vocabulary. It also struggled with queries that required more than one piece of information to be inferred, or that required reasoning beyond simple compositionality. Overall, the errors suggest that the model has limited ability to generalize to novel queries that deviate from the training distribution, and that further work is needed to improve its robustness and generalization capabilities.