Project 5 Report

Methods used:

I choose to implemenet the `Cover-Utt` method mentioned in the recommended paper. This method tries to find examples in the training to cover every word of the utterance.

The algorithm is as follows:

```
def diversity_prompts(utterance, examples):
    prompt_example = []
    for word in utterance:
        for example in examples:
            if word in example:
                prompt_example.append(example)
            examples.remove(example)

    prompt.construct(prompt_examples)
```

This algorithm gives an accuracy of 0.40. The result from uniform random sampling is about 0.325, and the embedding-based technique yields around 0.55. To get a better performance, instead of simply considering single word coverage, I also include two word sequence from the original utterance for coverage. This changes the third line of the algorithm to `for word[i] + word[i+1] in range(len(utterance)-1)`. This yields a slightly better accuracy of 0.45. I also tried to increase the number of consecutive words and found that the best performance happened when `n = 4`, i.e. we consider 4 words coverage first, then 3 words, etc. The final accuracy I got from this modified version of `Cover-Utt` gives me an accuracy of 0.65.

Error comparison:

The most interesting example I found is *which state borders the most states*, which translates to *(most (state , next_to_2 , state))*. However, in all the random prediction, embedding similarity prediction, and coverage-guided sampling using 3 or less consecutive words, the answer is *(largest_one (count_1 (next_to_2 (state)) , state))*, only the final version of `Cover-Utt` gives the right answer. This is because matching more words in the utterance yields more interesting and similar examples from the training set.