Task 1: Text Sentiment Classification

Comparison of two Bert models

I. Implementation

1. Load the transformers from hugging face
2. Decide use Bert or Roberta model to fine-tunel,
    i. Need to adjust the final linear layer which output features is 3 (positive, negative, neutral).
    ii. Fine tune the Roberta model need to change self.classifier to self.classifier.out_proj, else there might be some error.

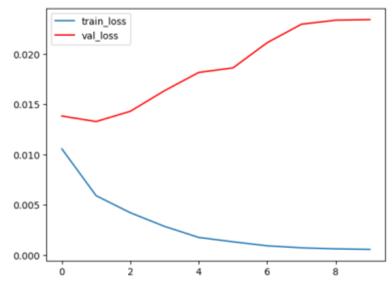
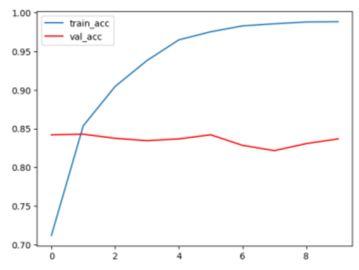
II. Performance:

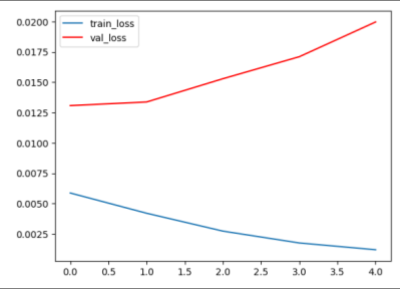
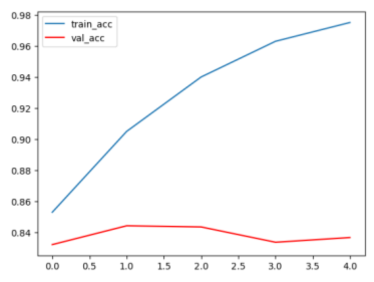
❿ **Bert-base-uncased: 0.8592**

❿ **Roberta-base: 0.8592**

❿ Bert with 5 epochs have lower accuracy (1e-2 lower, 0.84), while with 10 epochs training, two accuracies tend to have similar performance as Roberta.

- The reason could be that Bert model itself was trained with smaller amount of data in compared to Roberta model, thus the initial performance might not be good.
- There's also a saying that the loss function of Roberta model could be lower than Bert, as the chart showed below, Roberta train loss is much lower than Bert, and the validation loss of Bert goes much higher after more epochs.
- The accuracy of Bert model has a sharp increase on the beginning epochs, but stops after, while the Roberta model keep increase gradually.
- **A short conclusion is the Roberta can have slightly better performance than Bert model in terms of both loss and accuracy. While the result might not have significant performance.**

	Loss	Accuracy
Bert		

Roberta		

Task 2: In-context Learning

Different templates and verbalizers

Prompt1: Classify sentiment: It was a [MASK] emotion sentence.

[Positive, Negative, Neutral]

	Zero Shot	One Shot	Few Shot
Accuracy	0.236241	0.632221	0.437744
Precision	0.088943	0.548170	0.609918

[Favorable, Unfavorable, Neutral]

One-shot: accuracy 0.161397 | precision: 0.026092

One shot with prompt1 performed better on accuracy, few-shot got higher precision. The try of using different type of verbalizer performs quite bad at sentiment analysis. Reason could be that the verbalizer does not have quite straightforward demonstration on classifying sentiment.

Prompt2: Analyze sentiment: The emotion is [MASK].

[Positive, Negative, Neutral]

	Zero Shot	One Shot	Few Shot
Accuracy	0.205211	0.603435	0.526932
Precision	0.622626	0.563622	0.595807

One shot with prompt2 performed better on accuracy like above, while the overall precision value is similar. Few-shot performs better than prompt1 few-shot, could inference that this prompt is suitable for giving more examples.

Prompt3: The emotion sentence is [MASK].

[Positive, Negative, Neutral]

	Zero Shot	One Shot	Few Shot
Accuracy	0.679352	0.610851	0.526932
Precision	0.545498	0.471485	0.606658

The zero-shot prefix gives better performance. Which could tell that prompt3 is suitable for giving “What task to do” without showing too much example.

The three templates all have their pros and cons in terms of the accuracy and precision. Some prompt might be suitable for zero-shot while some prompt might be suitable for few-shot or one-shot. To choose the best prompt over these three, **prompt2** could be a better performance prompt in terms of its accuracy and precision.

Task 3: LM-BFF

Case1: Auto template and verbalizer

Best Loss: 0.9375, Last Loss: 0.75

Kaggle: 0.82131

Best template: *{"placeholder": "text_a"} It was {"mask"} . it was great.as lively an account as seinfeld is deadpan . It was awful. you 're not merely watching history , you 're engulfed by it . It was awesome.*

Best verbalizer: ['incredible', 'terrific']

The evaluation score of most auto template + auto verbalizer tests (many tries) tend to decrease after more epochs.

Case2: Manual template and auto verbalizer (three type of templates)

1. **Classify sentiment: It was [MASK: awful/awesome]**

   Best Loss: 0.96875, Last Loss: 0.9375

   Kaggle: 0.86721
2. **Classify sentiment: The emotion is [MASK: terrible/great]**

Best Loss: 0.9375, Last Loss: 0.9375

3. **Sentiment Analysis of the sentence: It is [MASK: awful/awesome]**
   Best Loss: 0.9062, Last Loss: 0.9062
4. **Sentiment is [MASK: awful/awesome]**
   Best Loss: 0.875, Last Loss: 0.6562
5. **Sentiment is classified to [MASK: awful/awesome]**
   Best Loss: 0.9375, Last Loss: 0.9062

Manual templates that I'd tried, generally, "Classify sentiment" template had a better performance. And the Kaggle result is even better than the auto generated template. In compare the 1. Longer sentences, 2. "classify" to "analysis", 3. words position changed, the short and easy "What task to do" and "The result" gives the better result.

While compared to auto generated templates result, which the auto templates "A [MASK]", "it was [MASK]", "It was [MASK]" had shown, but it could not improve performance that much with only 5 epochs. So, I think that this could be due to the model not able to generate more generalized template compared to manual template.

Another try was that I only used auto generated verbalizer that change both "candidate_num" and "label_word_num_per_class" to 40, and use manual template The performance on Kaggle is 0.9000 [MASK: Awesome, Terrific], which is much better than using both auto template generation and auto verbalizer generator.

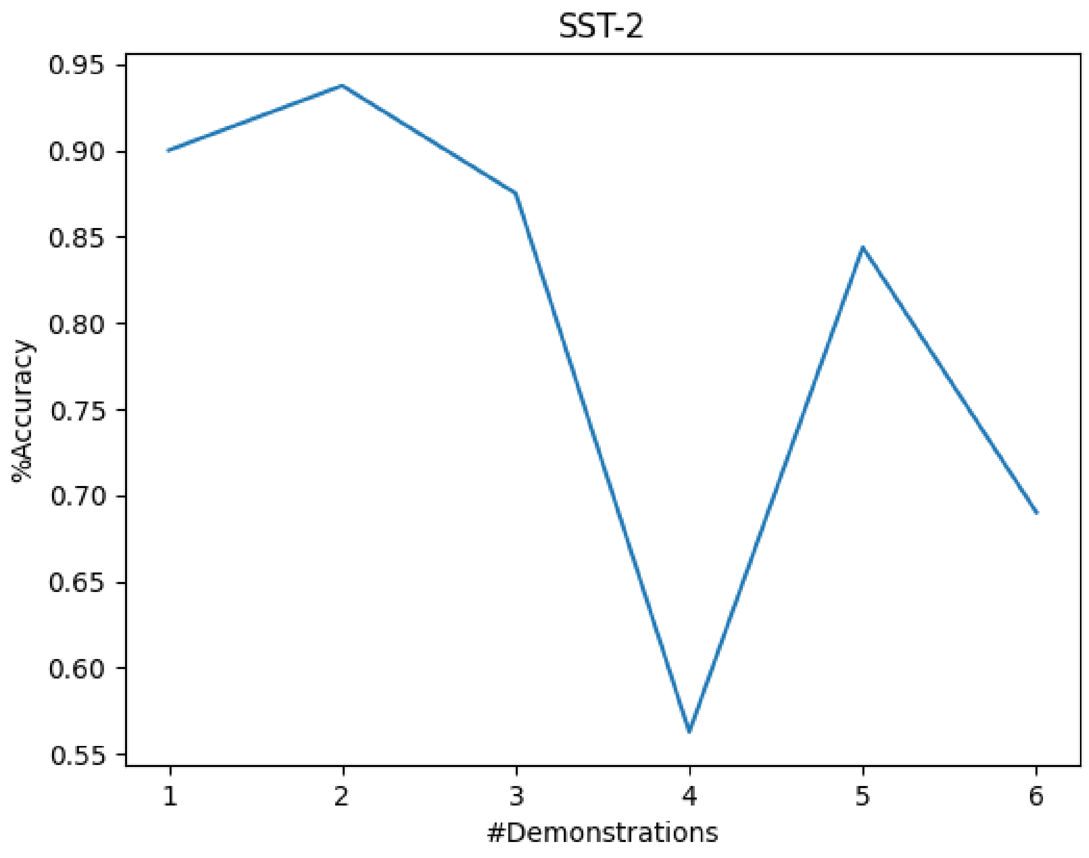
Short conclude on this part: ***Manual template + Auto verbalizer > Manual template + Manual verbalizer > Auto template + Auto verbalizer.***
However, I think if the increase on beam_width or the demonstrations could also improve the performance.

Case3: Different amount of demonstrations

Manual Prompt: Classify sentiment: It was [MASK: Awful/Awesome]

#Demonstrations	**Auto_t**	**Auto_v**	**Score**
2	**-**	**-**	**0.9**
3	**-**	**-**	**0.875**
4	**-**	**-**	**0.5**
5	**-**	**-**	**0.5**
6	✓ **(not finished)**	**-**	**0.69**

From the 3 demonstrations eval score show, the increase amount of demonstration didn’t increase the performance, which it not like the paper. The reason could be that I didn’t use “Auto verbalizer” as the verbalizer, or the gap between there demonstrations number is too small in compared to the paper (K=16, 32, 128, 256). However, the memory of Colab seems to be not enough or due to some bugs. So if the memory is accessible, I think the performance could be much better.


SST-2
%Accuracy
#Demonstrations