`

Task 1: Comparative Analysis of RoBERTa and BERT for Twitter Sentiment Classification

Ren-Di Wu

whats2000mc@gmail.com

1. Introduction:

This report details a comparative study of *roberta-base* and *bert-base-cased* models for classifying sentiments in the *twitter_sentiment* dataset, There are 10248 train samples, 1317 validation samples, and 3075 test samples.

The aim is to understand how these models perform in a three-label sentiment classification task.

2. Methodology:

Both models were adapted to classify text into three categories. The key implementation steps included:

- Adjusting the final layer of each model to accommodate three output labels.
- Setting the batch size to 32 for BERT training and batch size to 24 for RoBERTa training.
- Adapt the learning rate to 1e-5 with weight decay 0.01
- Employing a customized loss function, LabelSmoothingLoss, as described in this paper, rather than the traditional cross-entropy loss.
- Train and Test process by apply attention mask and forward to the model.

3. Results:

The results are presented in Table 1, showcasing the accuracy of both models across different label smoothing levels for validation and test datasets. (Table 1)

4. Analysis & Interpretation

The comparative analysis reveals distinctive insights into the performance of RoBERTa and BERT models with label smoothing. Notably, RoBERTa with LS3 achieves the highest accuracy on the test set at 0.87035, indicating an optimal balance between confidence and generalization.

BERT peaks at LS4 with a test accuracy of 0.85223, suggesting a different sensitivity to label smoothing. These results underscore the importance of hyperparameter tuning tailored to each model's characteristics.

5. Conclusion

The study concludes that label smoothing significantly enhances model performance, with RoBERTa marginally outperforming BERT in the test scenarios. The degree of label smoothing required for optimal accuracy varies between models, underscoring the necessity for model-specific hyperparameter optimization in

`

sentiment classification tasks on Twitter data.

6. Appendices

- Table 1: The results are showcasing the accuracy of both models
- Figure 1: Model setup code snippets.
- Figure 2: Label Smoothing Loss code snippets.

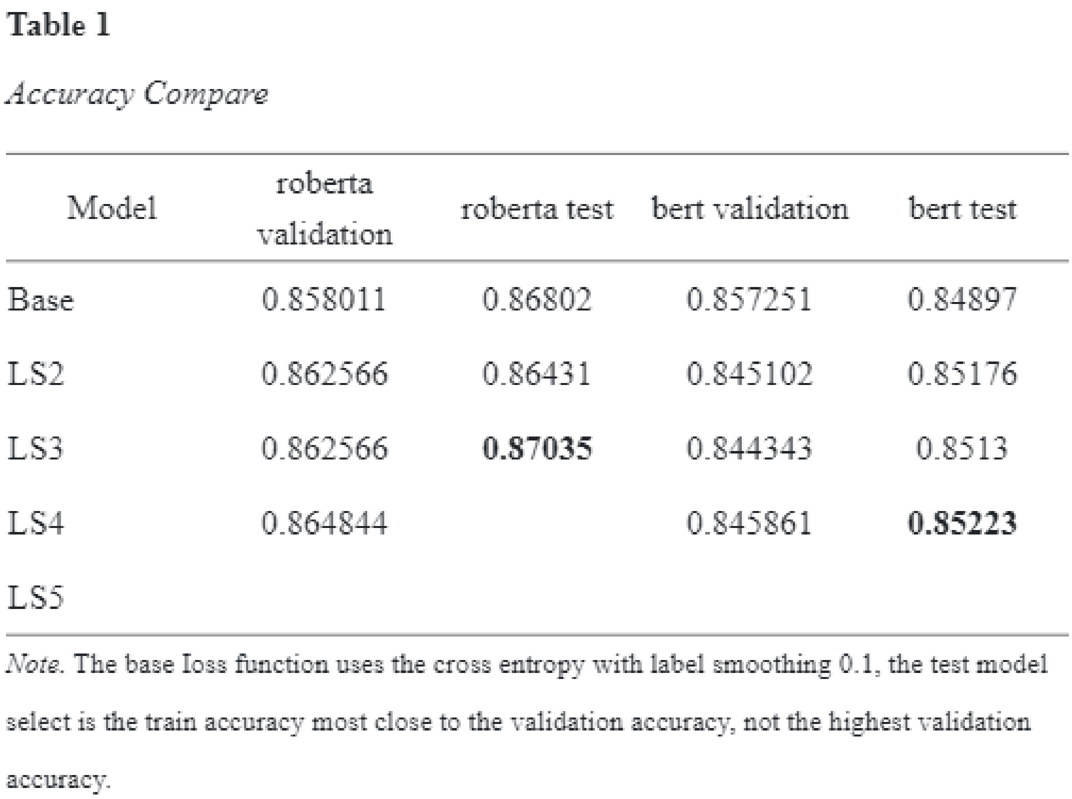


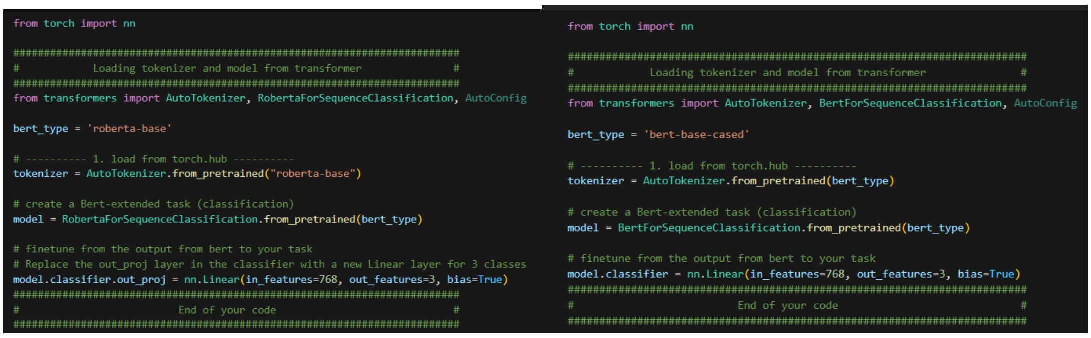
Table 1

Accuracy Compare

Model	roberta validation	roberta test	bert validation	bert test
Base	0.858011	0.86802	0.857251	0.84897
LS2	0.862566	0.86431	0.845102	0.85176
LS3	0.862566	**0.87035**	0.844343	0.8513
LS4	0.864844		0.845861	**0.85223**
LS5				

Note. The base loss function uses the cross entropy with label smoothing 0.1, the test model select is the train accuracy most close to the validation accuracy, not the highest validation accuracy.

^ Table 1: The results are showcasing the accuracy of both models

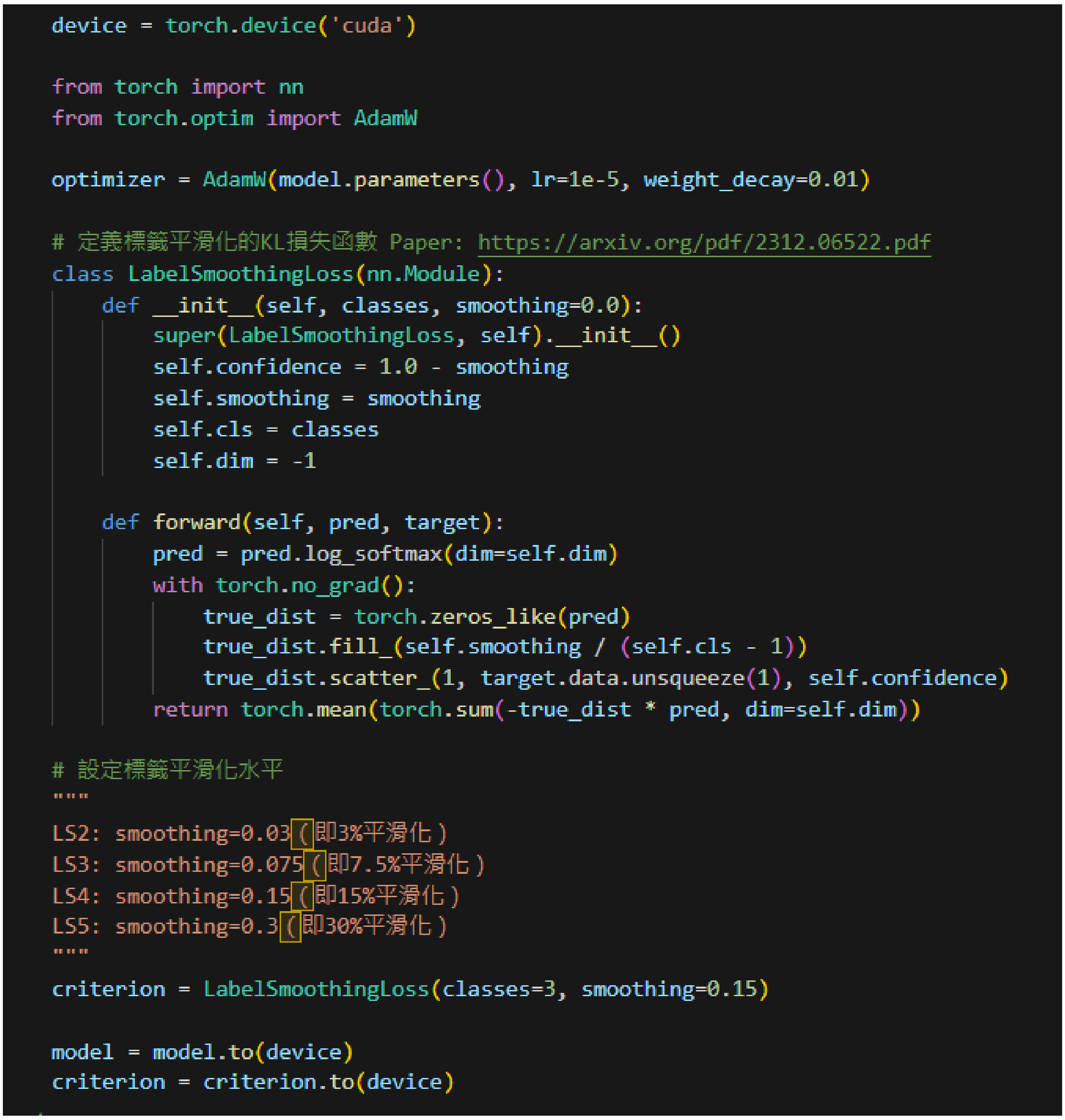


```
from torch import nn

##########################################################################
#            Loading tokenizer and model from transformer                #
##########################################################################
from transformers import AutoTokenizer, RobertaForSequenceClassification, AutoConfig

bert_type = 'roberta-base'

# ---------- 1. load from torch.hub ----------
tokenizer = AutoTokenizer.from_pretrained("roberta-base")

# create a Bert-extended task (classification)
model = RobertaForSequenceClassification.from_pretrained(bert_type)

# finetune from the output from bert to your task
# Replace the out_proj layer in the classifier with a new Linear layer for 3 classes
model.classifier.out_proj = nn.Linear(in_features=768, out_features=3, bias=True)
##########################################################################
#                           End of your code                             #
##########################################################################
```

```
from torch import nn

##########################################################################
#            Loading tokenizer and model from transformer                #
##########################################################################
from transformers import AutoTokenizer, BertForSequenceClassification, AutoConfig

bert_type = 'bert-base-cased'

# ---------- 1. load from torch.hub ----------
tokenizer = AutoTokenizer.from_pretrained(bert_type)

# create a Bert-extended task (classification)
model = BertForSequenceClassification.from_pretrained(bert_type)

# finetune from the output from bert to your task
model.classifier = nn.Linear(in_features=768, out_features=3, bias=True)
##########################################################################
#                           End of your code                             #
##########################################################################
```

^ Figure 1: Model setup and tokenizer for Roberta and Bert

`



```
device = torch.device('cuda')

from torch import nn
from torch.optim import AdamW

optimizer = AdamW(model.parameters(), lr=1e-5, weight_decay=0.01)

# 定義標籤平滑化的KL損失函數 Paper: https://arxiv.org/pdf/2312.06522.pdf
class LabelSmoothingLoss(nn.Module):
    def __init__(self, classes, smoothing=0.0):
        super(LabelSmoothingLoss, self).__init__()
        self.confidence = 1.0 - smoothing
        self.smoothing = smoothing
        self.cls = classes
        self.dim = -1

    def forward(self, pred, target):
        pred = pred.log_softmax(dim=self.dim)
        with torch.no_grad():
            true_dist = torch.zeros_like(pred)
            true_dist.fill_(self.smoothing / (self.cls - 1))
            true_dist.scatter_(1, target.data.unsqueeze(1), self.confidence)
        return torch.mean(torch.sum(-true_dist * pred, dim=self.dim))

# 設定標籤平滑化水平
"""
LS2: smoothing=0.03（即3%平滑化）
LS3: smoothing=0.075（即7.5%平滑化）
LS4: smoothing=0.15（即15%平滑化）
LS5: smoothing=0.3（即30%平滑化）
"""
criterion = LabelSmoothingLoss(classes=3, smoothing=0.15)

model = model.to(device)
criterion = criterion.to(device)
```

^ *Figure 2: Label Smoothing Loss*

`

Task 2a: Impact of Prompts on BERT Base Model Performance for Sentiment Classification

1. Introduction:

This section will introduce the experiment's main objective: to analyze how different templates and verbalizers influence the performance of the `bert-base- uncased` model in a sentiment classification task.

2. Methodology:

The experiment tests the performance of the *bert-base-uncased* model using a 3x3 matrix of templates and verbalizers. The first template is outlined as 'The sentence is [MASK].', with a corresponding verbalizer mapping sentiments to numerical values. This process will be replicated with two additional templates and two other sets of verbalizers, creating nine unique prompt configurations. The model configuration remains constant across tests with a maximum sequence length of 512 and a batch size of 32.

3. Results:

The outcomes of the experiment, given the intricate nature of the data, have been comprehensively tabulated. Tables 1 through 4 in the Appendices delineate the accuracy, precision, recall, and F1 scores for each permutation of template and verbalizer used in the model's evaluation. These tables provide a clear and detailed view of the performance impacts and allow for a nuanced comparison across the various prompt configurations. (Table 1 ~ 4)

4. Analysis & Interpretation

Our analysis shows that the choice of verbalizers significantly impacts the BERT model's performance in sentiment analysis. The verbalizer set to 'negative', 'neutral', and 'positive' achieved the highest scores, likely due to the unambiguous sentiment representation these terms provide.

They leading to clearer learning patterns for the model. In contrast, words like 'terrible', 'okay', and 'great' are less definitive, potentially leading to lower performance metrics.

This underscores the necessity of selecting verbalizers that offer clear and consistent sentiment distinctions to improve model accuracy.

The template 'Sentence for emotion: This sentence is [MASK].' was the most effective across verbalizers, indicating the influence of syntactic structure on the model's predictive capabilities.

5. Conclusion

The experiment underscores the significance of prompt design in sentiment analysis with BERT models. Choosing the right verbalizer and template syntax plays a crucial role in model performance. Specifically, using

`

clear sentiment descriptors as verbalizers and a sentence structure that closely mimics natural language resulted in higher accuracy, precision, recall, and F1 scores.

6. Appendices

The Appendices contain comprehensive tables that detail the results of the experiment.

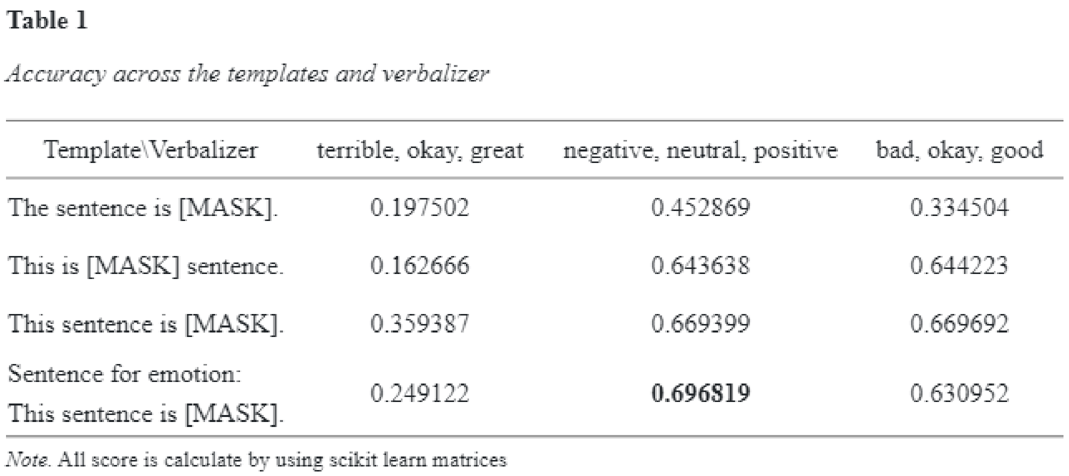


Table 1

Accuracy across the templates and verbalizer

Template\Verbalizer	terrible, okay, great	negative, neutral, positive	bad, okay, good
The sentence is [MASK].	0.197502	0.452869	0.334504
This is [MASK] sentence.	0.162666	0.643638	0.644223
This sentence is [MASK].	0.359387	0.669399	0.669692
Sentence for emotion: This sentence is [MASK].	0.249122	**0.696819**	0.630952

Note. All score is calculate by using scikit learn matrices

^ Table 1: The accuracy across the templates and verbalizer

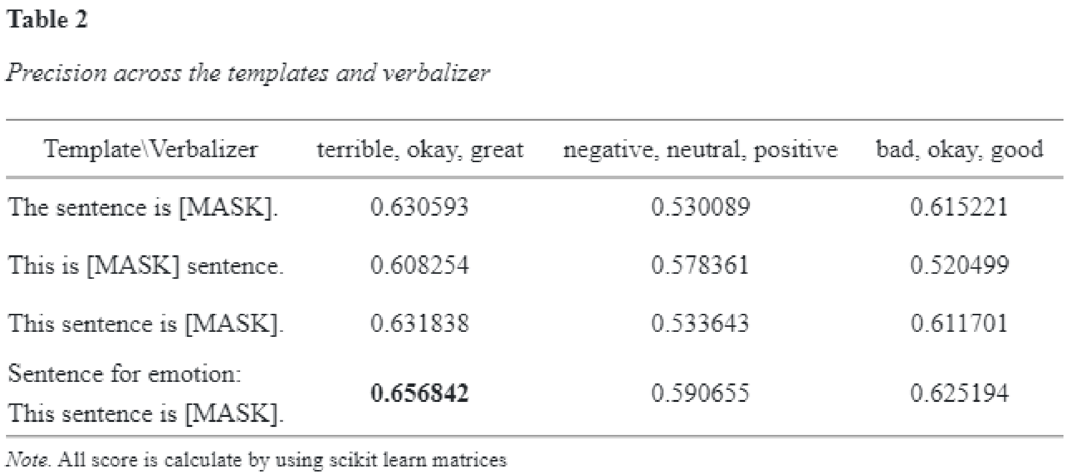


Table 2

Precision across the templates and verbalizer

Template\Verbalizer	terrible, okay, great	negative, neutral, positive	bad, okay, good
The sentence is [MASK].	0.630593	0.530089	0.615221
This is [MASK] sentence.	0.608254	0.578361	0.520499
This sentence is [MASK].	0.631838	0.533643	0.611701
Sentence for emotion: This sentence is [MASK].	**0.656842**	0.590655	0.625194

Note. All score is calculate by using scikit learn matrices

^ Table 2: The precision across the templates and verbalizer

`

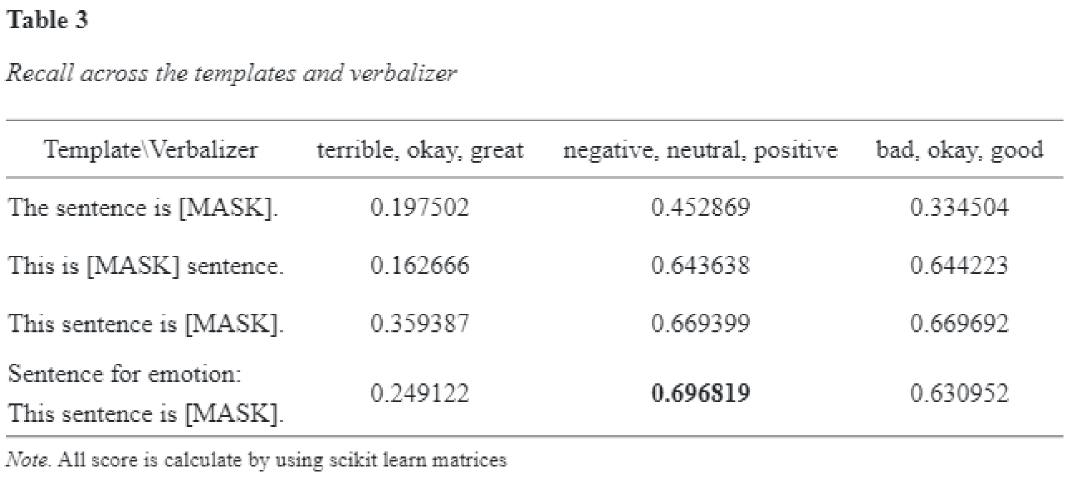


Table 3

Recall across the templates and verbalizer

Template\Verbalizer	terrible, okay, great	negative, neutral, positive	bad, okay, good
The sentence is [MASK].	0.197502	0.452869	0.334504
This is [MASK] sentence.	0.162666	0.643638	0.644223
This sentence is [MASK].	0.359387	0.669399	0.669692
Sentence for emotion: This sentence is [MASK].	0.249122	**0.696819**	0.630952

Note. All score is calculate by using scikit learn matrices

^ Table 3: The recall across the templates and verbalizer

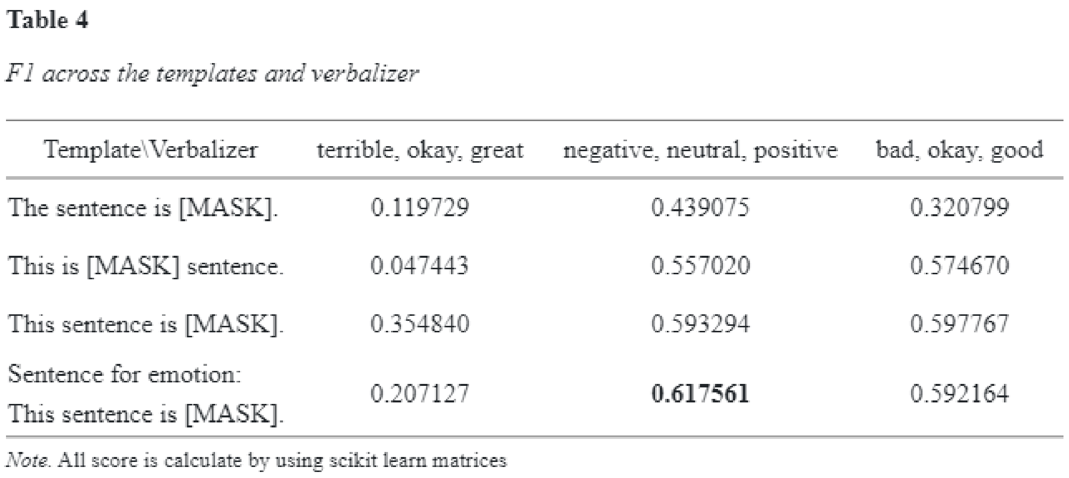


Table 4

F1 across the templates and verbalizer

Template\Verbalizer	terrible, okay, great	negative, neutral, positive	bad, okay, good
The sentence is [MASK].	0.119729	0.439075	0.320799
This is [MASK] sentence.	0.047443	0.557020	0.574670
This sentence is [MASK].	0.354840	0.593294	0.597767
Sentence for emotion: This sentence is [MASK].	0.207127	**0.617561**	0.592164

Note. All score is calculate by using scikit learn matrices

^ Table 4: The F1 score across the templates and verbalizer

`

Task 2b: Comparative Performance Analysis in Zero-Shot, One-Shot, and Few-Shot Learning with BERT

1. Introduction:

This report delves into an experiment comparing the best template and verbalizer combinations from Task 2a across zero-shot, one-shot, and few-shot learning scenarios using the BERT model. The focus is to assess the adaptability and learning efficiency of the model under varying levels of training data availability.

2. Methodology:

The top three template and verbalizer combinations identified in Task 2a were evaluated. Performance metrics were recorded in:

- Zero-shot learning (no training examples).
- One-shot learning (a single training example).
- Few-shot learning (a limited set of training examples).

3. Results:

The detailed results of the experiment, including accuracy, precision, recall, and F1 scores for zero-shot, one-shot, and few-shot scenarios with the three template variations, are presented in tables in the Appendices.

4. Analysis & Interpretation

The experiment highlights notable differences in model performance across learning scenarios:

- The template Sentence for emotion: This sentence is [MASK]. demonstrated the highest accuracy in the zero-shot scenario. While it showed a decline in the one-shot scenario, its performance remained relatively stable in the few-shot scenario, suggesting robustness across different levels of training data exposure.
- *This is [MASK] sentence.* demonstrated moderate performance in zero-shot and one-shot but significantly dropped in few-shot learning.
- This sentence is [MASK]. performed consistently well in zero-shot, with a slight decrease in accuracy in one-shot and few-shot scenarios.

5. Conclusion

The findings suggest that the BERT model's ability to adapt to sentiment analysis tasks varies significantly with the amount of training data and the structure of the prompt. While certain templates perform better in a zero-shot scenario, their effectiveness can vary in one-shot and few-shot learning. This underscores the importance of prompt engineering and data availability in fine-tuning models for specific NLP tasks.

`

6. Appendices

The Appendices contain comprehensive tables that detail the results of the experiment.

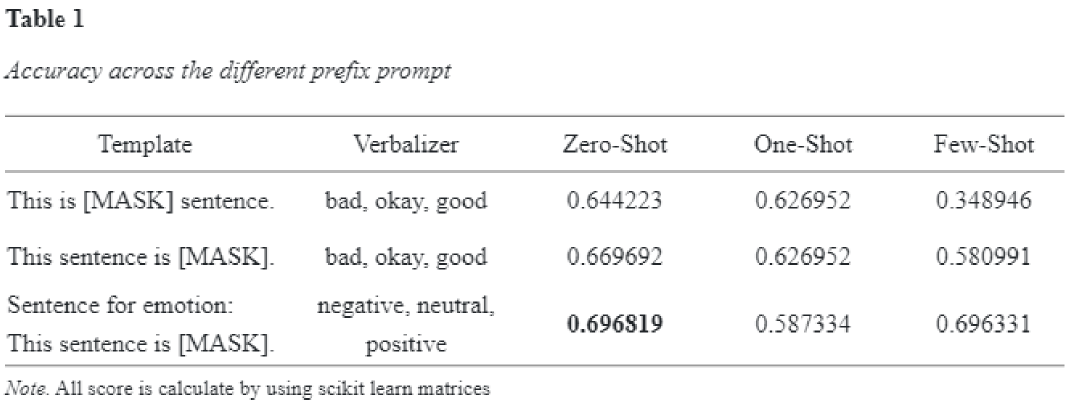


Table 1

Accuracy across the different prefix prompt

Template	Verbalizer	Zero-Shot	One-Shot	Few-Shot
This is [MASK] sentence.	bad, okay, good	0.644223	0.626952	0.348946
This sentence is [MASK].	bad, okay, good	0.669692	0.626952	0.580991
Sentence for emotion: This sentence is [MASK].	negative, neutral, positive	**0.696819**	0.587334	0.696331

Note. All score is calculate by using scikit learn matrices

^ *Table 1: The accuracy across the different prefix prompt*

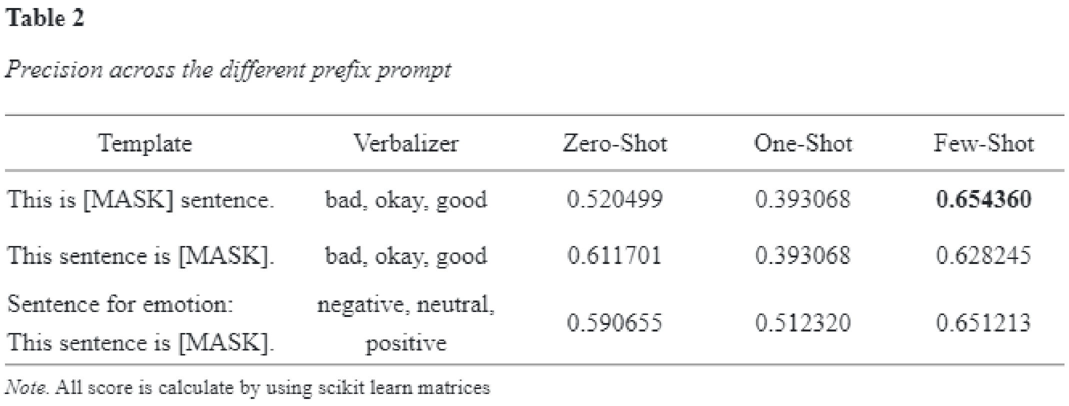


Table 2

Precision across the different prefix prompt

Template	Verbalizer	Zero-Shot	One-Shot	Few-Shot
This is [MASK] sentence.	bad, okay, good	0.520499	0.393068	**0.654360**
This sentence is [MASK].	bad, okay, good	0.611701	0.393068	0.628245
Sentence for emotion: This sentence is [MASK].	negative, neutral, positive	0.590655	0.512320	0.651213

Note. All score is calculate by using scikit learn matrices

^ *Table 2: The precision across the different prefix prompt*

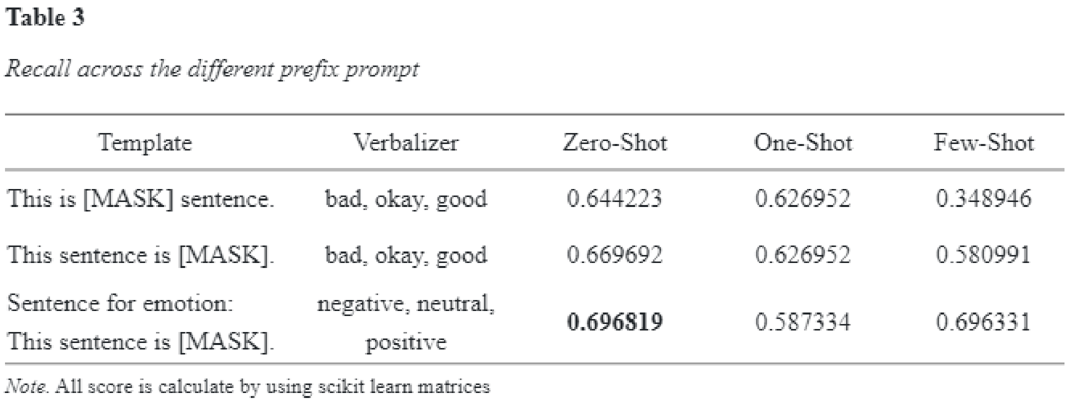


Table 3

Recall across the different prefix prompt

Template	Verbalizer	Zero-Shot	One-Shot	Few-Shot
This is [MASK] sentence.	bad, okay, good	0.644223	0.626952	0.348946
This sentence is [MASK].	bad, okay, good	0.669692	0.626952	0.580991
Sentence for emotion: This sentence is [MASK].	negative, neutral, positive	**0.696819**	0.587334	0.696331

Note. All score is calculate by using scikit learn matrices

^ *Table 3: The recall across the different prefix prompt*

`

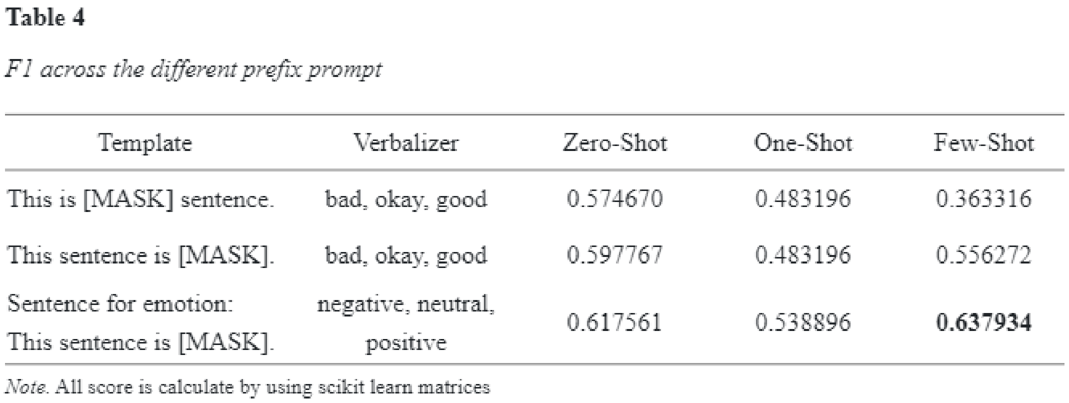


Table 4

F1 across the different prefix prompt

Template	Verbalizer	Zero-Shot	One-Shot	Few-Shot
This is [MASK] sentence.	bad, okay, good	0.574670	0.483196	0.363316
This sentence is [MASK].	bad, okay, good	0.597767	0.483196	0.556272
Sentence for emotion: This sentence is [MASK].	negative, neutral, positive	0.617561	0.538896	**0.637934**

Note. All score is calculate by using scikit learn matrices

^ *Table 4: The F1 score across the different prefix prompt*

References

[1] OpenAI. (2023). ChatGPT [Large language model]. https://chat.openai.com

[2] Making Pre-trained Language Models Better Few-shot Learners. https://arxiv.org/pdf/2012.15723.pdf

[3] Label Smoothing for Enhanced Text Sentiment Classification. https://arxiv.org/pdf/2312.06522.pdf

[4]