COMP4332/RMBI4310 Project 1

Group 14

Shenyue LI slidl@connect.ust.hk
Xuanrong GAO xgaobb@connect.ust.hk
Zijian YE zyeaw@connect.ust.hk
Ziruo XIAO zxiaoal@connect.ust.hk

1 Introduction

Our team has developed a sentiment analysis tool for restaurant reviews that predicts ratings on a scale of 1 to 5 stars for both the validation and test sets. By utilizing various models and feature extraction techniques during the training phase, we were able to achieve enhanced prediction accuracy. The assessment of the model's effectiveness will entail the utilization of the Macro-F1 score derived from the validation set as the primary metric, with a predetermined threshold of 0.5360 serving as the baseline. Through experimentation with varying mechanisms, our team was able to surpass this established benchmark and achieve a Macro-F1 score of 0.66 through the implementation of our developed model.

2 Data Analysis

The original files consist of three datasets: training (18,000 records), validation (2,000 records), and testing (4,000 records). Our study focuses on the relationship between "text" and "stars," while the correlation between different labels such as "cool," "useful," and "funny" and stars, and the frequency of different labels in each star category do not show significant evidence of aiding in training and prediction. Thus, we excluded these variables from the model (refer to Appendix 2).

3 Methodology

3.1 TF-IDF and Classifiers

In this subsection, we designed pipeline models, combining feature engineering processes and classification.

In feature extraction, we introduced Term Frequency Inverse Document Frequency (TF-IDF) to tackle the unbalanced problem in the raw data. TF-IDF extracts and weights informative words and transforms textual data into a numerical representation with less noise for future classification.

In the classification process, we employed three basic classification models, Logistic Regression(LR), Support Vector Machines(SVM), and eXtreme Gradient Boosting(XGBoost), where XGBoost is a scalable tree boosting system, to classify the inputted dataset.

3.2 BERT

Our team opted to utilize the powerful BERT pre-trained model for our project. BERT is a machine learning model based on transformers that applies an attention mechanism to learn contextual relations between words. This approach enables BERT to achieve exceptional performance on NLP tasks by capturing the nuances of language usage and sentence structure.

3.3 RoBERTa

RoBERTa is a pre-trained language model developed by Facebook for NLP tasks. We adopted this model to improve upon the performance of the BERT by utilizing a larger corpus of text data and a modified training approach (Liu et al., 2019). RoBERTa is highly effective for fine-tuning on specific downstream tasks with relatively small amounts of task-specific data.

4 Experiment

4.1 Environment Setup

In this project, we utilize Google Colab and its free GPU to train advanced model, BERT and RoBerta. The GPU is Tesla T4 with 15G RAM source. To conduct training process, Python 3.9 and PyTorch3 2.0.0+cu118 are involved.

4.2 Data Preprocessing

- **Stemming and stopword removal**: Text preprocessing was conducted to reduce related words to their root form, and uninformative English words (e.g., 'is', 'and', etc.) were removed to minimize vocabulary size before feeding into the training mechanism. Such techniques proved crucial for feature engineering techniques such as TF-IDF.
- **Adding beginning and ending tokens**: In our BERT model, this step incorporates sentence structure by helping BERT identify the beginning and the end of input texts, thereby enabling it to process fixed-length chunks of text.
- **Creating an attention mask**: This helps mask irrelevant tokens so BERT can focus on important expressions, specifically sentiment-bearing phrases in

our case. It also prevents attention allocation to extra padding, thereby improving flexibility.

- **Adjusting the maximum length of the texts**: By tuning the maximum sequence length, we aim to determine the optimal value for maximizing performance and robustness when processing diverse inputs. A few different truncation and padding strategies are tested.

4.3 Model Training

4.3.1 TF-IDF and classifiers

To implement the pipelines, we first imported the TfidfVectorizer function from sklearn to employ TF-IDF for preprocessing the text data. We adjusted the parameters to remove accents, perform NFKD normalization, and converted all characters to lowercase before tokenizing during the preprocessing step.

To implement the Logistic Regression model, we employed LogisticRegression from sklearn, setting the inverse of the regularization strength to 1 for stronger regularization, and utilized a combination of L1 and L2 regularization while controlling its balance. To optimize this pipeline, we employed saga, a solver algorithm for large datasets, and assigned different weights to different classes for a balanced approach.

For the Support Vector Machines model, it came from svm in sklearn. We utilized a linear kernel to transform input data and balanced the input dataset using the same strategy as the logistic regression model above.

For eXtreme Gradient Boosting (Chen & Guestrin, 2016), we import XGBClassifier from xgboost module and use 25 trees with small depth and child weight. To prevent overfitting, we randomly sample each tree and extracted input data and apply L1/L2 regularization.

Classifier	Macro-F1	Accuracy	Better ?
LR	0.56	0.62	√
SVM	0.54	0.62	√
XGBoost	0.45	0.58	×

4.3.2 BERT

We first load the BERT base uncased model and worked on fine-tuning the model based on hyperparameters suggested by the paper (Albanese, 2022).

After several initial attempts, we achieved a maximum validation accuracy and Macro-F1 score around 0.67 and 0.59 respectively without further improvement. We then applied multiple data preprocessing techniques mentioned in the previous session and in the class(apart from those required by BERT), but the performance turned out to have little improvement. Worth noting, we later figured out two specialized strategies which turned out to be quite beneficial to the final result:

- **Learning Rate:** Adjust the learning rate during training. Set it large at the beginning and smaller in later iterations.

- **Proportion of data in different labels:** After noticing that the model achieved good performance for stars 1 and 5 with F1-scores around 0.8, we observed that its performance for stars 2, 3, and 4 was much worse, particularly for stars=2. This is likely due to the imbalanced proportion in the training data and the inherent difficulty of these cases, as customer attitude is relatively ambiguous. To address this issue, we conducted large-scale training on a normal training set for many iterations to enhance the overall accuracy. Then, we trained on a smaller training set with an increased proportion of 2, 3, and 4 stars for several iterations to improve the F1-score for these labels. We continuously saved the best-performing intermediate model and continued training on it.

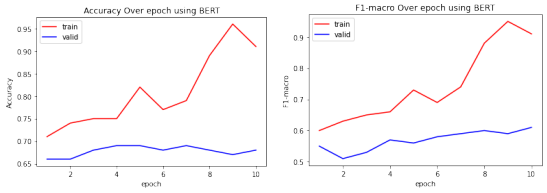
Attempts are listed in the Appendix.

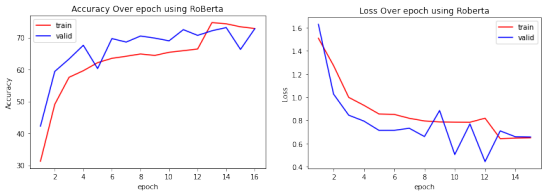
4.3.3 RoBERTa

We have implemented comparable strategies for RoBERTa (Taunk, 2021) as we did for BERT, and the details of our attempts are provided in the Appendix.

5 Results

The statistics for our best-performing BERT and RoBERTa models are presented below. In terms of the validation set, we achieved the best accuracy and Macro-F1 scores of 0.69 and 0.61 for BERT, and 0.73 and 0.66 for RoBERTa, respectively. It is evident that the latter model outperformed BERT in all criteria, resulting in a more satisfactory performance.





The performance of RoBERTa for each label is presented below. It can be observed that the model achieved scores above 0.5 for stars 2, 3, and 4 (i.e. label 1, 2, 3 shown in the image below).

accuracy for validation set

	precision	recall	f1-score	support
0	0.84	0.81	0.82	292
1	0.46	0.56	0.51	163
2	0.61	0.48	0.54	232
3	0.62	0.53	0.57	421
4	0.82	0.90	0.86	892
accuracy			0.73	2000
macro avg	0.67	0.66	0.66	2000
weighted avg	0.73	0.73	0.73	2000

```
[[236  53   3   0   0]
 [ 35  92  32   2   2]
 [  5  49 111  54  13]
 [  3   6  33 223 156]
 [  2   1   3  81 805]]
Accuracy on test data = 73.35%
```

6 Discussion and Conclusion

Due to the poor performance observed in stars 2, 3, and 4, we propose an idea of merging 2 and 3 stars into a single label and subsequently conducting a 2-class classification to better differentiate them. This idea may be further developed in the future.

In conclusion, we implemented five distinct mechanisms to perform sentiment analysis and compared their Macro-F1 scores. Our results indicate that RoBERTa was the best fit model, yielding a Macro-F1 score of 0.66.

7 Reference

Albanese, N. C. (2022, May). Fine-tuning bert for text classification. Medium.
Retrieved from https://towardsdatascience.com/fine-tuning-bert-for-text-classification-54e7df642894

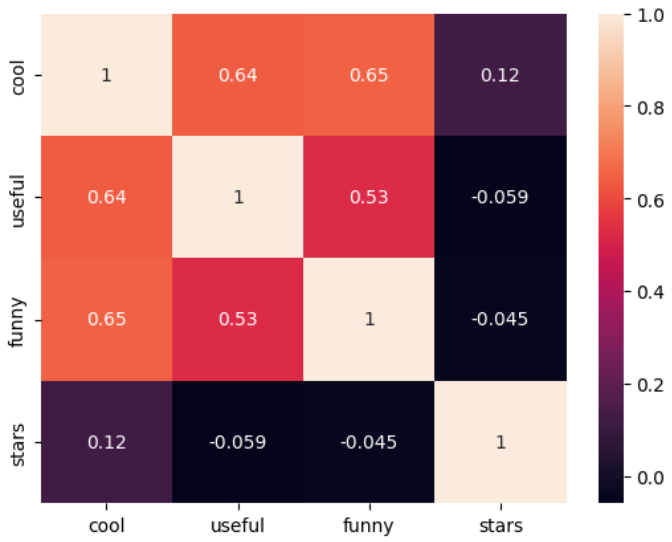
Chen, T., Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

Taunk, D. (2021). NLP_scripts [GitHub repository]. Retrieved from https://github.com/DhavalTaunk08/NLP_scripts

8 Appendix

1. Correlation between "funny", "cool", "useful" and "Stars" after removing all the "0" value



Stars	1	2	3	4	5
Cool	575	591	1451	3283	4556
useful	5209	2219	2926	5314	8545
funny	1593	960	1367	2136	2673

2. TF-iDF

The images below from the left to the right are for TF-IDF + Logistic Regression, TF-IDF + Support Vector Machines and TF-IDF + eXtreme Gradient Boosting

	precision	recall	f1-score	support
1	0.68	0.80	0.73	292
2	0.35	0.41	0.38	163
3	0.39	0.46	0.42	232
4	0.45	0.46	0.46	421
5	0.83	0.71	0.77	892
accuracy			0.62	2000
macro avg	0.54	0.57	0.55	2000
weighted avg	0.64	0.62	0.62	2000

```
[[234  41  10   5   2]
 [ 51  67  38   4   3]
 [ 21  48 107  44  12]
 [  8  22  88 194 109]
 [ 32  12  33 183 632]]
accuracy 0.617
```

	precision	recall	f1-score	support
1	0.68	0.78	0.73	292
2	0.33	0.34	0.34	163
3	0.36	0.42	0.39	232
4	0.44	0.51	0.48	421
5	0.85	0.71	0.78	892
accuracy			0.62	2000
macro avg	0.53	0.55	0.54	2000
weighted avg	0.64	0.62	0.62	2000

```
[[228  44  12   3   5]
 [ 57  56  41   6   3]
 [ 23  44  98  57  10]
 [ 12  12  90 215  92]
 [ 14  13  28 203 634]]
accuracy 0.6155
```

	precision	recall	f1-score	support
0	0.67	0.64	0.65	292
1	0.30	0.15	0.20	163
2	0.39	0.28	0.33	232
3	0.38	0.31	0.34	421
4	0.67	0.85	0.75	892
accuracy			0.58	2000
macro avg	0.48	0.44	0.45	2000
weighted avg	0.55	0.58	0.55	2000

```
[[186  16  18  19  53]
 [ 44  24  15  36  44]
 [ 20  18  65  72  57]
 [ 12  13  46 131 219]
 [ 16  10  21  89 756]]
accuracy 0.581
```

3. BERT

Single-training set pipeline for BERT

Train set size	Batch Size	Learning Rate	Max length	epoch	F1-Marco
5000	16	3e-5	64	2	0.59
5000+1492	16	3e-5	128	1	0.57
5000+1492	16	3e-5	64	1	0.58
7000	32	2e-5	64	2	0.59

Multi-training set pipeline for BERT

epoch	Train set size	Batch Size	Learning Rate	Max length
1	10300	32	5e-5	64
2	4200	32	2e-5	64
3	4200	32	2e-5	64
4	1878 (227:392:170:353:736)	32	1e-5	64
5	1748(239:190:257:361:701)	32	1e-5	64
6	1748(239:190:257:361:701)	32	1e-5	64
7	962(130:128:127:224:353)	32	1e-5	64
8	756(90:156:116:136:258)	32	1e-5	64

== validation accuracy = 0.68 == validation F1-Marco = 0.61

4. RoBERTa

Single-training set pipeline for RoBERTa

Train set size	Train Batch Size	Valid Batch Size	Learning Rate	Max length	epoch	F1-Marco
10800	32	4	1e-5	256	1	0.61
12600	32	4	2e-5	256	3	0.63
14400	32	4	2e-5	256	1	0.58
18000	8	4	1e-5	64	2	0.64
10800	32	4	5e-5	256	2	0.65

Multi-training set pipeline for RoBERTa

Epoch	Train data	Train Batch Size	Learning Rate	Max length
First epoch	trainset[:9000]	32	3e-5	256
Second epoch	trainset[9000:14000] 2 stars data in trainset[:3000] 3 stars data in trainset[:600] 4 stars data in trainset[:300]	32	5e-5	256
Third epoch	trainset[:10800]	32	2e-5	256
Last epoch	trainset[14000:16000] all the data with stars from 2 to 4	16	1e-5	256

== validation accuracy = 0.73 == validation F1-Marco = 0.66