Project 1 Sentiment Analysis Report

Group 4

Liu Dingdong, Wang Weiqi, Wang Yiren, Zhou Zixuan

1. Explorative Dataset Analysis

There are three files in the dataset: training, validation, and test set, whose length ratio is 9:1:2. After training the models on the training set, we will use the validation set to select the best model in terms of Macro-F1. And test its performance on the testing set. Each set's data volume and average review length are shown in table 1.

	Number of data	Avg review length
Training set	18000	132.843
Validation set	2000	127.654
Testing set	4000	130.503

Table 1 Statistics of three data splits

We selected the "text" and "stars" columns in the dataset. The "text" column includes review sentences with textual sentiments such as "love" and "hate" to reveal the reviewer's attitude. For the "stars" column, five unique sentiments can be found in the dataset, ranging from 1 (lowest score) to 5 (highest score). Important statistics on the training dataset are presented in Table 2. From our observation, data with star 5 make up most of the dataset, with the shortest average review length. Meanwhile, the dataset has the least data with star 2, which has the longest average review length.

Stars	Number of data	Avg review length
1	2672	168.153
2	1457	171.042
3	1989	157.838
4	3971	136.600
5	7911	105.712
All	18000	132.843

Table 2 Statistics of five sentiments

2. Methodology

2.1 BERT (base, distilled)

BERT is short for Bidirectional Encoder Representation from Transformers. It is designed to pre-train deep bidirectional representations from the unlabeled text by jointly conditioning on both left and right contexts in all layers. To explore the impact of different versions of BERT, we experimented with two versions of BERT in this project: BERT-base and BERT-distilled. These models have different configurations of transformer blocks and thus achieve different performances. For example, the BERT-base has 12 transformer layers with 12 attention heads and 10M total parameters. BERT-distilled is a distilled version of BERT designed to be smaller, faster, cheaper, and lighter. It has 40% fewer parameters than BERT-base and runs 60% faster, preserving over 95% of BERT's performance.

2.2 BART

BART is a denoising autoencoder for pretraining sequence-to-sequence models. We adopted "bart-base" version from Facebook as the pre-trained language model. Hyperparameters of the model's setting were all set to the default values, such as 12 encoder and decoder layers, 16 attention heads, and 1024 dimensions in all the layers, including the pooler layers. In total, our BART model consisted of 139M parameters.

2.3 GloVe

GloVe stands for Global Vectors for Word Representation. It is an unsupervised learning algorithm for obtaining word embeddings. We used the GloVe pre-trained on 840 billion words, which outputs word embedding in the dimensionality of 300. Moreover, we used Spacy as the tokenizer. For each token unknown to GloVe, a randomly generated 300-dimension vector was assigned to it so that all tokens were covered in our word embedding bank. GloVe was effective since only 10.29% of the total 68,000 unique tokens were not covered. After retrieving the word embedding, we used a Bi-LSTM layer followed by a fully connected layer to classify each review sentence. Before passing to the network, all reviews were padded to the maximum review length (1209). Dropout and batch normalization layers were also added to prevent overfitting. In total, the model consisted of 12.7M of parameters, which was far less than other models.

3. Experiment

3.1 Setup

In this project, both the tokenizers and the pre-trained models were downloaded from the huggingface library[1]. Tokenizers were used to tokenize the texts, add special tokens, and generate attention masks. We used the uncased version for all the BERT and BART pre-trained models. Experiments were conducted on Linux GPU clusters with multiple GeForce RTX 3080Ti graphical cards installed. Each cluster also has 512G memory space. Anaconda [2] Python 3.8 was used as the developing environment, and PyTorch[3] 1.10.0 was used as the deep learning development framework.

For all the models, we used the AdamW optimizer with a default learning rate of $5 * 10^{-5}$. To avoid overfitting, we

[1] https://huggingface.co/

[2] https://www.anaconda.com/

[3] https://pytorch.org/

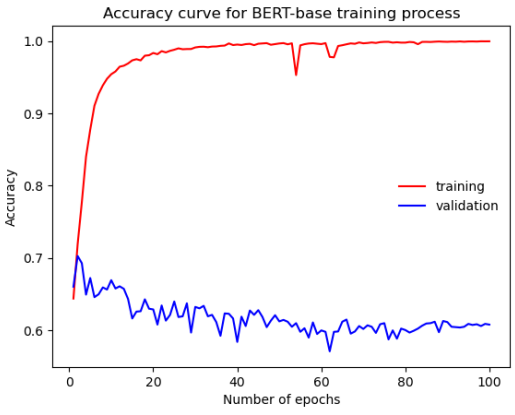
further added a linear scheduler to decrease the learning rate along the training process. All the models were trained with 100 epochs and a batch size of 10 to ensure consistency between different methodologies.

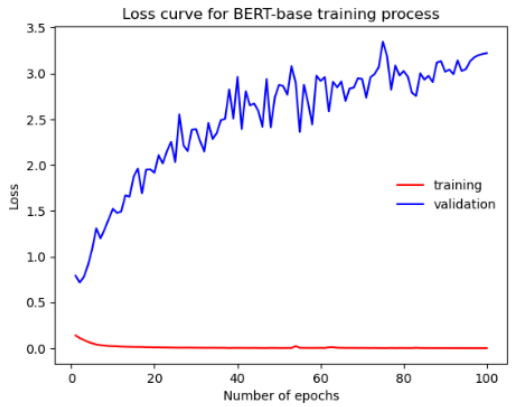
3.2 Result

	Macro-F1	Precision	Recall	Accuracy
Weak Baseline	*0.4270*	*0.5420*	*0.4325*	*0.6135*
Strong Baseline	*0.5370*	*0.5509*	*0.5324*	*0.6500*
BERT-base	*0.6308*	*0.6300*	*0.6300*	*0.6840*
BERT-distilled	*0.6230*	*0.6200*	*0.6300*	*0.6905*
BART	*0.6140*	*0.6300*	*0.6100*	*0.6840*
GloVe	*0.5399*	*0.5600*	*0.5500*	*0.6325*

Table 3 Evaluation Statistics of All Models

Table 3 presents the macro statistics of all our experimented models and the baselines. BERT-base achieved the best performance in terms of macro-F1. While different versions of BERT achieved similar results, the macro-F1 score by BART was slightly lower than the BERT model series, and GloVe's was much lower than other language models empowered classifiers. Nevertheless, all methods achieved better performance than the strong baseline. The figures below plot the BERT-base model's loss curve and accuracy curve concerning the number of training epochs to analyze our training process.





3.3 Analysis

3.3.1 Observing the Accuracy and Loss Curve

According to the Accuracy Curve, as the number of epochs increases, the training accuracy increases significantly (after only 10 epochs) while the validation accuracy continually decreases. Further training does improve the model's performance in validation. As for the Loss Curve, the training loss remains very low while validation loss keeps increasing despite fluctuation along the curve. Both trends indicate that the model is overfitting. When the model encounters new text, the past training data does not help it correctly predict reviewers' sentiments.

One possible explanation for this is that the model is over-trained. We might have used too many epochs during training, which confines the language model to the context of training data and makes it generate bias when extracting features in the validation set.

3.3.2 Model Comparison: BERT vs. BART

According to our experiment result, BERT outperforms BART in all evaluation metrics. This proved that BART was particularly effective when fine-tuned for text generation tasks but is weaker than BERT in text comprehension tasks. In this project, BERT is more suitable for encoding the text and extracting the text's features to predict the sentiments than BART. One potential reason is that BART used a left-to-right decoder (like GPT) instead of a bidirectional decoder (like BERT), making BART more competitive at text generation tasks but not feature extraction.

3.3.3 Model Comparison: BERT vs. GloVe

Notably, the performance of GloVe is significantly weaker than the BERT models and the BART model. This indicates that static word representation methods were considerably less effective than dynamic contextualized word representations methods. When encoding the text, GloVe encodes the sentence by considering every single word, while BERT relies on the attention mechanism to generate high-quality context-aware word embeddings. Embeddings are refined during the training process by passing through each BERT encoder layer. Word embeddings are positionally encoded to keep track of the pattern or position of each word in a sentence. As of GloVe, only the Bi-LSTM and FC layers parameters are trained, while the embeddings remain the same. This resulted in a poorer performance compared with dynamic language models.

4. Conclusion

In this project, we implemented four different models to conduct sentiment analysis. By comparing their performance, we found the best model is BERT-based.