Sentiment Analysis Using Deep Learning

Ally Qin, Jingshu Lu, Wenya Xie, Zhaohua Huang

1. Introduction

Opinion mining becomes essential for business intelligence. According to statistics [1], Amazon ranks as the most popular shopping app with 150.6 million mobile users in September 2019 and brought in $75.5 billion in sales revenue in the first quarter of 2020. In addition, Amazon seems to continue dominating the total apparel sales from Walmart, Target and other competitors that 70% of the apparel shoppers would consider Amazon as the primary marketplace to shop [2]. Amazon, one of the top ecommerce companies in the United States, requires sentiment analysis to fully understand the customers' feedback in order to perform better in the market and increase sales on its platform, especially in the apparel market. In this analysis, the women clothes review dataset is implemented with four different deep learning models: Bidirectional Encoder Representations from Transformers (BERT), RoBERTa, XLNet, bidirectional LSTM (BiLSTM), and ensemble models to gain understanding of the emotional tone, attitudes, and emotions from the given text. The evaluation metric is determined by the accuracy function of predictions and the actual labels; it is used for both confusion matrix and ROC (receiver operating characteristic) plotting. As the final conclusion, XLNet outperforms all the other models and has the highest validation accuracy and test accuracy.

1.1 Problem statement

Customers' feedback is important and useful if it is interpreted accurately. Any business is obliged to understand their customer's needs, opinions and their satisfaction with the product. However, large web-based companies, similar to Amazon, contain millions of opinions under one product, and having just two polarities (good or bad) for the pre-defined comment dictionary is not enough to interpret the actual meaning behind those words. For instance, if the model interprets the words from a comment based on keyword or rule-based extractions, misclassification of the sentence might occur; it could be a positive comment, but got translated with a neutral or even negative tone of voice. Sentiment analysis with deep learning models could help and advance the model and avoid these types of problems.

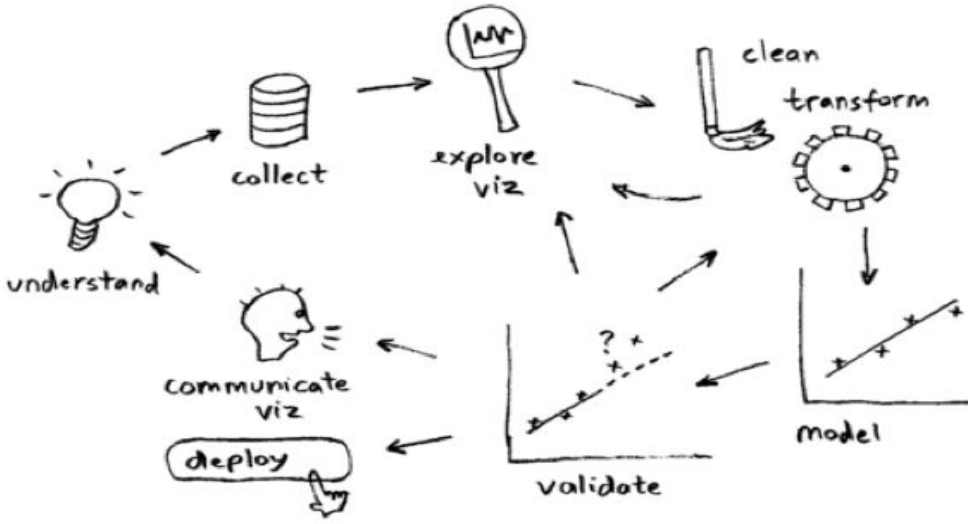
1.2 Applications

The purpose of using sentiment analysis is to classify the polarity of a given text at the document, sentence, feature, or aspect level for the subjective data. Also, whether the expressed opinion in a document, a sentence, an entity feature, or an entity aspect is positive, negative or neutral. It is useful in an ecommerce platform to monitor the customers' opinion about certain products and gain an overview of the universal trends. Such real-time monitoring capabilities makes Amazon have a quicker response towards customer's reaction, for both positive and

negative feedback. In addition, Amazon could take advantage of this competence to make improvements and invest in potential brands or markets that could satisfy customers' needs. In addition, the sentiment analysis should have a main focus on how to make the final results more interpretable and actionable for the business needs. For instance, customers' feedback could be a major factor for pricing plans, customer service, sponsorship, and brand plan developments. Therefore, the company is able to obtain clearer directions to make changes.

1.3 Approach project workflow

The project workflow follows a cycle: problem understanding, data collection, data visualization, clean and transform, modeling, validation, deployment, and communication. During the first stage of the process, it is required to fully understand the problem statement and define approaches from different perspectives, as well as, comprehend the objectives and goals. The second stage of the process is data collection; collect initial data, provide background information of the dataset, describe the data, and understand the attributes. The third stage of the process is exploratory data analysis; using tools to do basic visualization, perform simple statistical analysis on the dataset, and verify data quality. The fourth stage of the process includes cleaning the dataset by dropping the nulls, replacing the missing values, and transforming the dataset by standardization, one-hot encoding, etc. When this stage is finished, it could either go back to the third stage to visualize the cleaned dataset or move onto the next step. The fifth stage of the process is modeling; select modeling technique, generate test design, build model parameter settings, and assess model. During the sixth stage of the process is validation; evaluate results by using predefined metrics and select the model with the highest accuracy. If the result from this stage is not satisfying, models could either be retrained or go back to the third stage to restart the same process. During the final stage of the process, deployment phase is reached; determine the monitoring and maintenance plan, and establish a report for business insight presentation.



2. Data preparation

2.1 Dataset Collection

The dataset used in this study came from Kaggle – Women's E-Commerce Clothing Reviews[3]. This dataset contains 23,486 rows and 10 features. This dataset consists of reviews written by real customers, hence it has been anonymized, i.e. customer names were not included, and references to the company were replaced with 'retailer'.

2.2 Data Exploration & Visualization

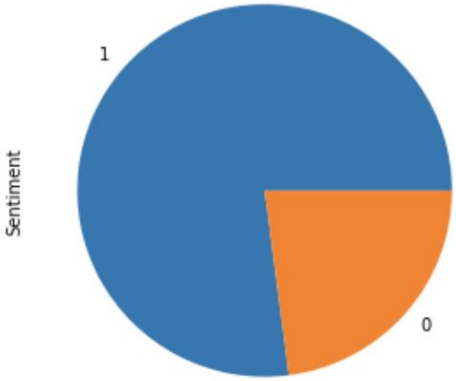
Since our sentiment analysis is based on review text and rating score. We only kept these two columns and dropped the other. We first checked null values in our dataset, then deleted rows containing null values, then we converted the rating score to binary value: 1indicated positive review with rating score larger than 3, 0 indicates negative review with rating score otherwise. Below shown the dataset after cleaning and statistical description of this dataset. After all, we got 22,641 data.

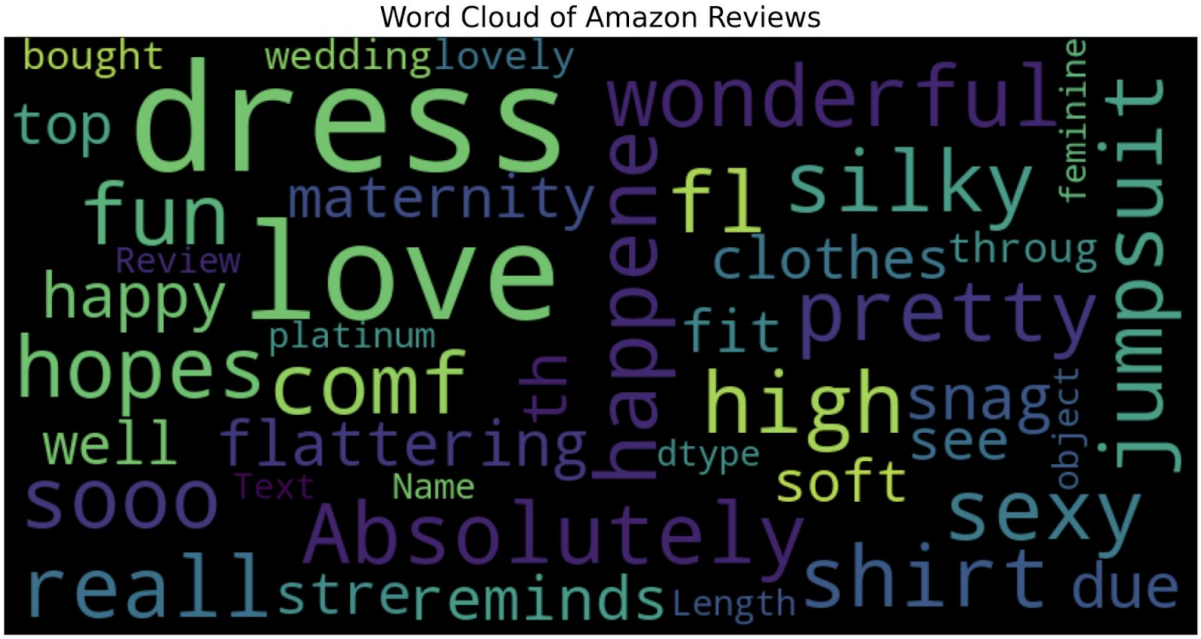
	Text	Sentiment
0	Absolutely wonderful - silky and sexy and comf...	1
1	Love this dress! it's sooo pretty. i happene...	1
2	I had such high hopes for this dress and reall...	0
3	I love, love, love this jumpsuit. it's fun, fl...	1
4	This shirt is very flattering to all due to th...	1

	Rating	Sentiment
count	22641.000000	22641.000000
mean	4.183561	0.770637
std	1.115762	0.420432
min	1.000000	0.000000
25%	4.000000	1.000000
50%	5.000000	1.000000
75%	5.000000	1.000000
max	5.000000	1.000000

The Pie chart showed the distribution of positive and negative reviews. There were 2 thirds of positive reviews and 1 third of negative ones.

Word cloud gave us the most frequent words in a review text. Most of the words are shown as positive review, but this does not necessarily indicate that the entire product review has a positive sentiment. Take note that word cloud only accounts for the frequency of words, and does not account for phrases.





2.3 Data Transformation

1) Prepare BERT/RoBerta/XLnet input

In our study, Bert, Roberta and XLnet used their mode base tokenization. BERT and RobBerta use special tokens CLS and SEP (mapped to ids 101 and 102) standing for the beginning and end of a sentence. XLnet uses CLS and SEP as well, but has CLS placed at the end of sentences. After getting these tokens, we convert them to integers, these integers are the unique ids corresponding to each token. The reviews in our dataset have varying lengths. To train the

models, the input sentence should be of equal length. So we padded the shorter sentences by 0s. The segment_id is used to distinguish between two sentences. In our study, we used 0s for BERT and RoBerta, 2s for XLnet. Below shows the input format feeded into the BERT Model.

Tokens:['[CLS]', 'i', 'saw', 'this', 'online', 'first', 'and', 'didn', '"', 't', 'care', 'for', 'it', '.', 'then', 'i', 'saw', 'it', 'in', '-', 'store', 'and', 'purchased', 'it', 'with', 'little', 'memory', 'of', 'the', 'online', 'experience', ',', 'because', 'the', 'dress', 'is', 'that', 'much', 'better', 'in', 'person', '[SEP]']

Input_ids:[101, 1045, 2387, 2023, 3784, 2034, 1998, 2134, 1005, 1056, 2729, 2005, 2009, 1012, 2059, 1045, 2387, 2009, 1999, 1011, 3573, 1998, 4156, 2009, 2007, 2210, 3638, 1997, 1996, 3784, 3325, 1010, 2138, 1996, 4377, 2003, 0, 0, 0, 0, 0, 0]

Input_mask:[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]

Segment_ids:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Label_id:1

2) Prepare BiLSTM input

In BiLSTM mode, we first declared fields for text and label, then we used pertain GloVe embedding, which were trained on 6 billion tokens and the embedding are 100 dimensional to build vocabulary. After this we padded the text to be the same length to process in batch. Below shows the input format feeded into the BiLSTM Model.

Field: {'text': ['i', 'received', 'this', 'romper', 'last', 'week', 'and', 'have', 'already', 'worn', 'it', 'twice', '.', 'it', 'is', 'soooo', 'comfortable', 'and', 'the', 'material', 'is', 'really', 'soft', '.', 'the', 'romper', 'is', 'loose', ',', 'yet', 'fitted', 'and', 'very', 'cute', '.', 'i', 'love', 'the', 'open', 'back', 'and', 'the', 'shorts', 'are', 'just', 'the', 'right', 'length', '.', 'i', 'wish', 'there', 'were', 'other', 'patterns', 'or', 'colors', 'available', ',', 'too', '.', 'i', 'would', 'definitely', 'buy', 'again', '!'], 'label': '1'}

TEXT.build_vocab, LABEL.build_vocab = dict_keys(['text', 'label'])

Text : ['i', 'received', 'this', 'romper', 'last', 'week', 'and', 'have', 'already', 'worn', 'it', 'twice', '.', 'it', 'is', 'soooo', 'comfortable', 'and', 'the', 'material', 'is', 'really', 'soft', '.', 'the', 'romper', 'is', 'loose', ',', 'yet', 'fitted', 'and', 'very', 'cute', '.', 'i', 'love', 'the', 'open', 'back', 'and', 'the', 'shorts', 'are', 'just', 'the', 'right', 'length', '.', 'i', 'wish', 'there', 'were', 'other', 'patterns', 'or', 'colors', 'available', ',', 'too', '.', 'i', 'would', 'definitely', 'buy', 'again', '!']

Label: '1'

3. Model Selection

In our project, we tried four state-of-art NLP models to perform sentiment analysis and compared their classification results based on the same splitted dataset. In addition, we created our ensemble model through soft voting. The four models we used are BERT, Roberta, XLNet and BiLSTM. For the first three models which are derived from Transformers, we leveraged transfer learning to fine-tune them to meet our requirements; for the last model, we spent more time training it from scratch. The detailed explanation of each model and ensemble method is as follows.

3.1 BERT

BERT stands for Bidirectional Encoder Representations from Transformer, which reads the entire sequence of tokens at once. The original Transformer has an encoder for reading inputs and a decoder for making predictions, but Bert just uses the encoder part. Different from RNNs, Bert is non-directional and its attention mechanism allows it for learning contextual relations

between words. Bert was trained on a large amount of unlabeled textual data by masking around 15% of the tokens with the goal to guess them. Instead of assigning the model a specific target, the masking method enables it to remember the contextual representation for every input token, leading to a better performance.

Since Bert is pre-trained and has learned sufficient language representations, we leveraged transfer learning to fine-tune it for our sentiment classification task. We used the pre-trained model in Pytorch Transformer API, tweaked the original model to predict the class label and returned the predicted class with a larger value. We customized the input sequence length to 128 to fit the length of most reviews in our training set. We tried several combinations of hyperparameters and optimizers, and finally found that batch size 8, learning rate 2e-5 and optimizer Adam, produced a better result on the validation set. In order to avoid overfitting and gradient exploding, we applied max norm regularization in the training process, clipping values larger than 1. Bert is a complex deep learning model that requires a lot of computing resources and training time. Hence we only trained it with three epochs using Colab. Surprisingly, even with a little amount of fine-tuning, Bert achieved over 90% accuracy on the test set.

3.2 RoBERTa

Roberta is an advanced version of Bert. It robustly optimized Bert by retraining it with improved training methodology, 10 times more data and computational power. To improve the training procedure, Roberta introduced dynamic masking so that the masked token changes during the training epochs, which makes the model more robust to the variations of inputs. Larger batch-training sizes were also found to be more useful in the training procedure. Through these modifications, Roberta usually outperforms Bert on similar tasks.

For model training, we adopted the same training method, the transfer learning, as Bert. Pre-trained Roberta model is also available in Pytorch Transformer API, and we tweaked its output layer to perform the binary classification and tried different hyperparameter combinations. The final combination we chose is batch size 8, learning rate 2e-5, epoch 3, optimizer Adam, and we introduced max norm regularization in this network as well.

3.3 XLNet

XLNet is a large bidirectional transformer that uses improved training methodology, larger data and more computational power to achieve better performance than Bert. To improve the training, XLNet introduces permutation language modeling, where all tokens are predicted but in random order. This is in contrast to the traditional language models, where all tokens were predicted in sequential order. Permutation modeling helps the model to learn bidirectional relationships and therefore better handles dependencies between words. In addition, the most powerful Transformer XL was used as the base architecture, which showed good performance even in the absence of permutation-based training.

The pre-trained XLNet is also accessible in Pytorch Transformer API, and we decided to follow the same method of training Bert and Roberta to fine tune this model with three epochs. The hyperparameters we finally used are batch size 8, learning rate 2e-5, epoch 3, optimizer Adam, and regularization techniques were also applied.

3.4 BiLSTM

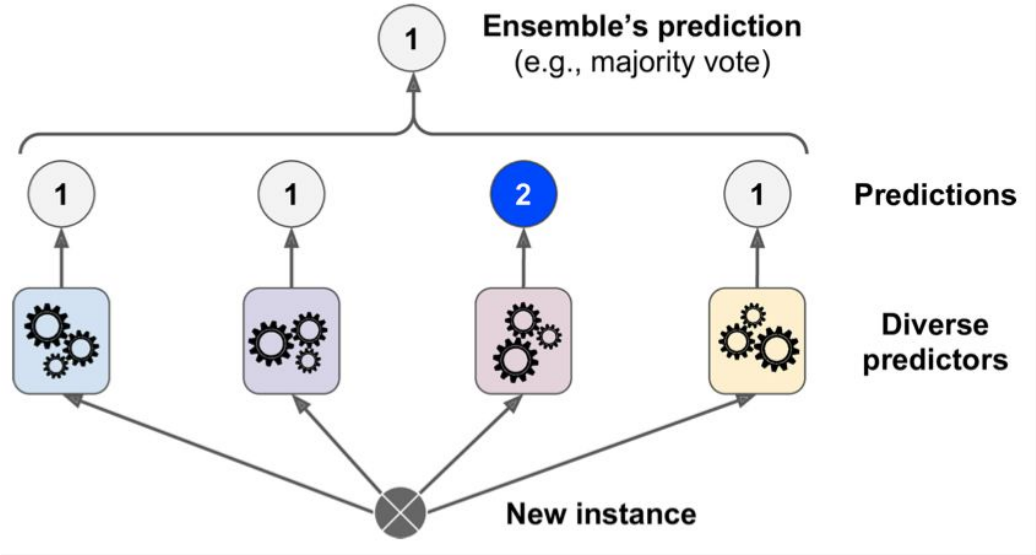
Bidirectional LSTM is a sequence processing model that consists of two LSTMs: one taking the input in a forward direction, and the other in a backwards direction. This architecture allows the model to know what words immediately follow and precede a word in a sentence, effectively increasing the amount of information available to the network and improving the context available to the algorithm. Since BiLSTM takes word embeddings as the input, we utilized Glove, a widely used unsupervised word embedding algorithm, to obtain the representation vectors of tokens in the training set first, and then fed those embedded vectors into BiLSTM for further training.

We used Pytorch to build our BiLSTM model and trained it from scratch through more epochs compared to the other three models. We used dropout in our network to cope with overfitting, and the dropout rate was set to 0.5. Besides, we tuned the hyperparameters in order to get better performance. According to the accuracy on the validation set, we finally chose batch size 128, epoch 25, learning rate 2e-2 and optimizer Adam.

3.5 Ensemble Models

In addition to the four individual models, we also implemented ensembles models :

- Soft-voting: We added a softmax layer to each individual model, making the sum of probability of the two classes equal to 1, and averaged the predicted probabilities given by these four models. Then we applied argmax to get the final class prediction of the ensemble model.
- Hard-voting: We count the predicted result of each individual model and output the majority selection of the classification result for each test case.



4 Model Evaluation

We divided the dataset into the training set, validation set and test set with the ratio of 50%, 25% and 25%. We compared the performance of each model based on the same validation set and test set. We also tried a traditional machine learning method, logistic regression, to do this task, and

used its prediction result as baseline. We tried to find out whether deep networks can outperform traditional methods with more complicated architecture but less feature engineering effort.

5. Results & performance

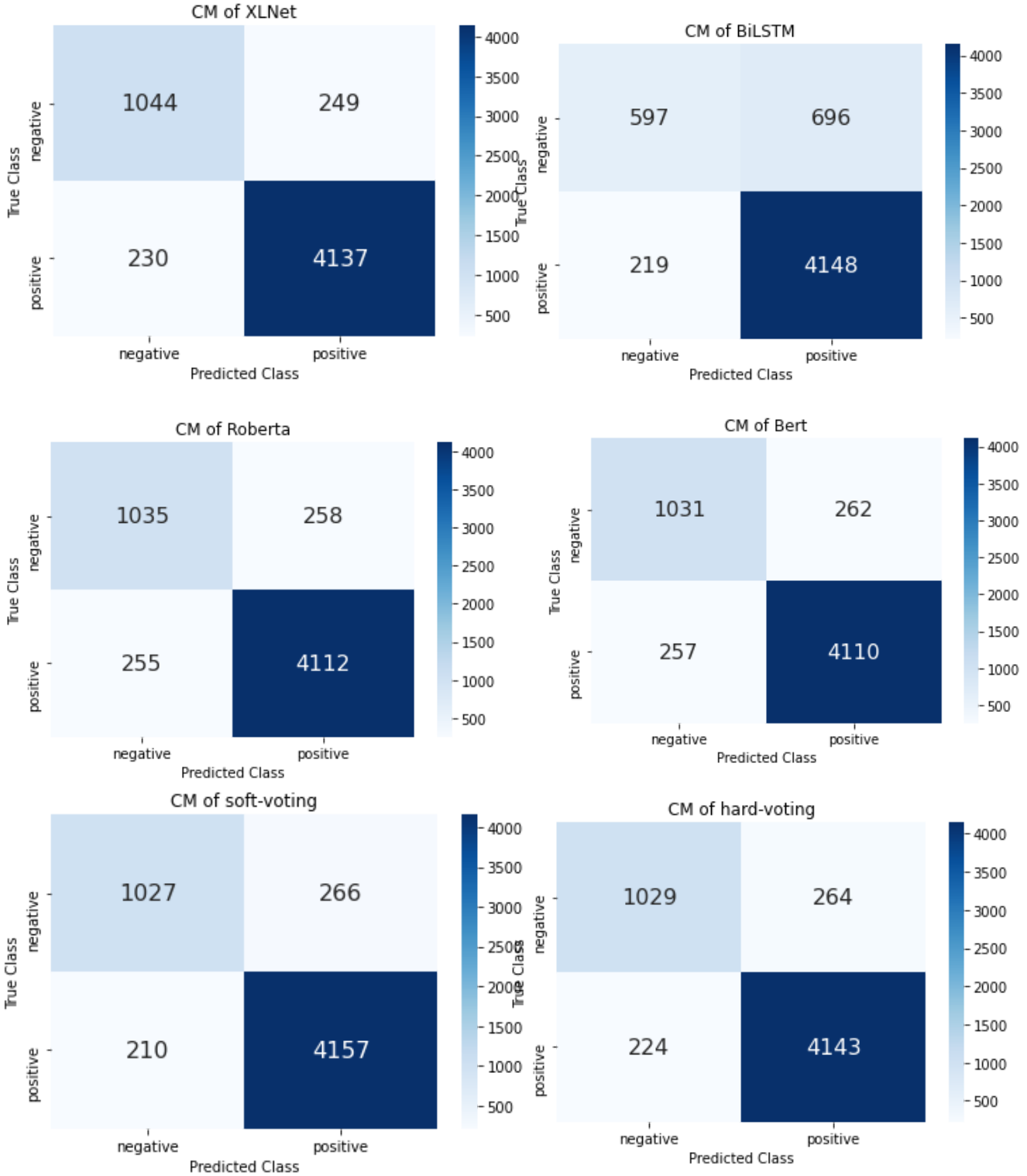
The prediction accuracy of each model is listed in below table:

Model	Bert	Roberta	XLNet	BiLSTM	Ensemble: Hard-vote	Ensemble: Soft-vote
Validation Accuracy	91.13%	91.4%	91.6%	76.83%	N/A	
Test Accuracy	90.58%	91.41%	91.73%	74.8%	91.4%	91.6%

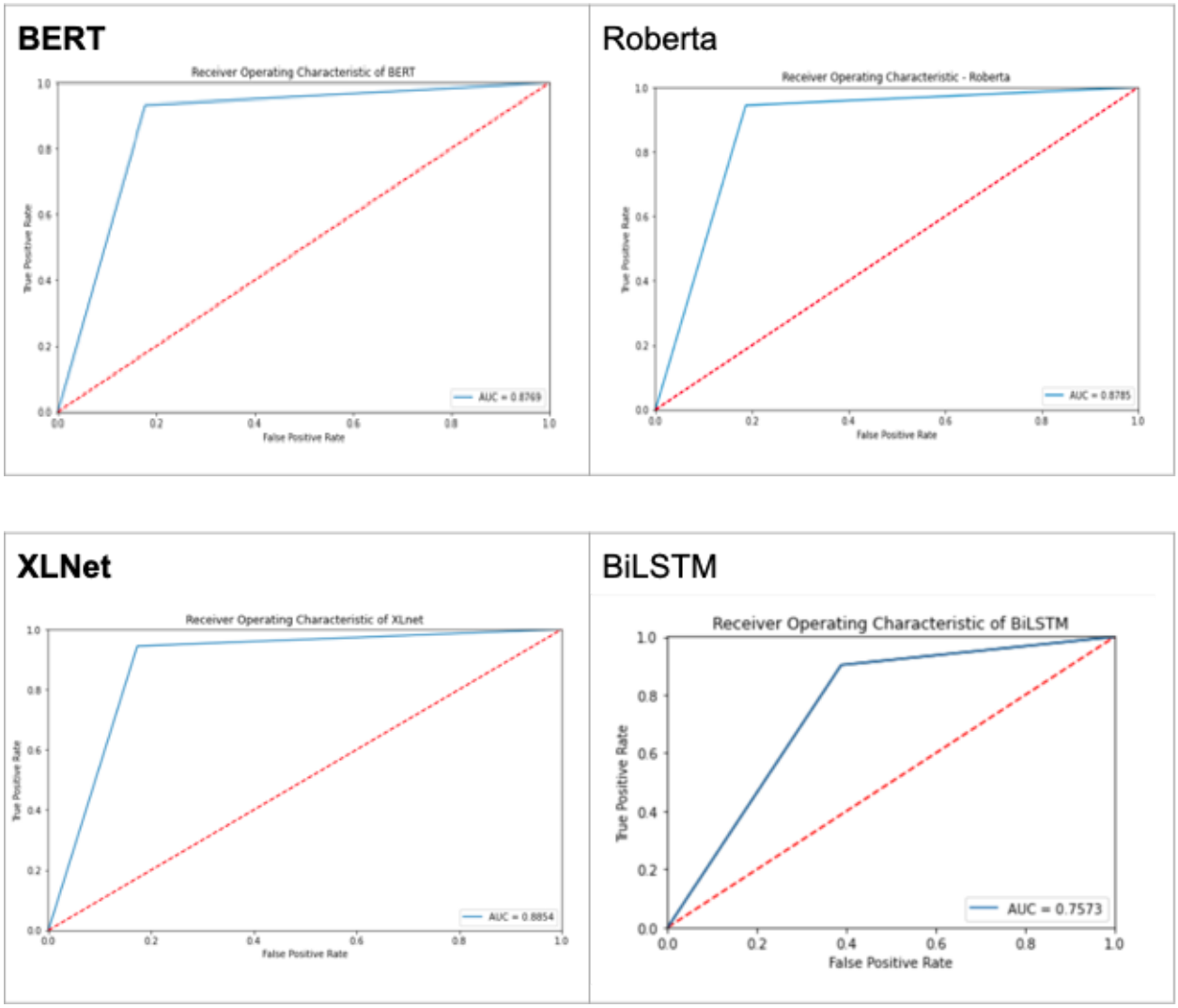
BiLSTM achieves the lowest accuracy. Then comes Bert and Roberta. And XLNet achieves the highest accuracy in both validation and test dataset. These results align with the models' design. Roberta is the fine-tuning based on Bert Model and XLNet also makes two improvements(1, The [MASK] token used in training does not appear during fine-turning;2,BERT generates predictions indepently) based on Bert Model. The later improved model does get better results.

Surprisingly, the two ensemble models don't outperform the XLNet model. Ensemble methods work best when the predictors are as independent from one another as possible. One way to get diverse classifiers is to train them using very different algorithms. This increases the chance that they will make very different types of errors, improving the ensemble's accuracy. But Bert, Roberta and XLNet have internal relationships. The diversities don't help in this case.

But when we have a close look at the Confusion Matrix, we can find the true positive prediction is improved with ensemble models. Especially with the soft-voting model, 4208 cases are predicted as positive correctly. The TPRs of soft-voting, hard-voting, Bert, Roberta, XLNet, Bi-LSTM are 95.2%, 94.9%,94.1%, 94.2% 94.7,95% respectively.


CM of XLNet
1044
249
230
4137
True Class
negative
positive
Predicted Class
CM of BiLSTM
597
696
219
4148
CM of Roberta
1035
258
255
4112
CM of Bert
1031
262
257
4110
CM of soft-voting
1027
266
210
4157
CM of hard-voting
1029
264
224
4143
4000
3500
3000
2500
2000
1500
1000
500

We use AUC-score to evaluate the model performance regardless of the threshold. Please see the AUC score and ROC curve of each model:



6. Conclusion

Using the advanced NLP models, we can analyze whether the customer feedback is positive or negative based on the review text. With Bert, Roberta and XLNet models we can achieve accuracy above 90% . These accuracy are reliable for product feedback analysis.

Reference

[1] M. Mohsin, “10 Amazon Statistics You Need to Know in 2020 [April 2020],” *Oberlo*, 03-Dec-2020. [Online]. Available: https://www.oberlo.com/blog/amazon-statistics. [Accessed: 06-Dec-2020].
[2] M. O'Brien, “Amazon Increases its Dominance in Apparel Sales,” *Multichannel Merchant*, 21-Apr-2020. [Online]. Available:

https://multichannelmerchant.com/ecommerce/amazon-increases-dominance-apparel-sales/. [Accessed: 06-Dec-2020].
[3] Nicapotato, “Women's E-Commerce Clothing Reviews,” *Kaggle*, 03-Feb-2018. [Online]. Available: https://www.kaggle.com/nicapotato/womens-ecommerce-clothing-reviews. [Accessed: 07-Dec-2020].