Sentiment Analysis in Financial Text Data

DataHack Advance Track Final Report

Zixin Ma, Sheng Yang

1 Introduction

Sentiment analysis on text data has been widely employed in the finance industry for predicting stock returns. A positive sentiment signals a long position, or a purchase, and a negative sentiment indicates a short position, or a sell. A high accuracy of identifications of sentiments helps fund managers achieve great returns, thus sentiment analysis is gaining traction lately.

Recent advancement in sentiment analysis has been observed in the deep learning realm, where transformer-based deep learning models take precedence in performance. The strong predictive power is largely credited to the availability of a large pretrained model, giving great comforts to researchers in the sentiment analysis field.

In this project, we demonstrate the superiority of transformer-based deep learning models to non-deep-learning alternatives, and discuss possible tracks of improvements that are not fully explored due to a limited time.

2 Exploratory Data Analysis

The provided dataset contains one feature named **Sentence**, the texts to extract features from, and one label called **Sentiment**, containing three labels: *positive*, *negative*, and *neutral*. This amounts to a multi-label classification task.

Table 1: Basic Stats

stats	number
num_train_samples	4383
num_test_samples	1461
num_labels	3

A strike class imbalanced is observed, where the majority of the labels are *neutral*. This suggests the necessity to split train dataset using a stratified scheme to ensure label distributions are the same in training set and validation set.

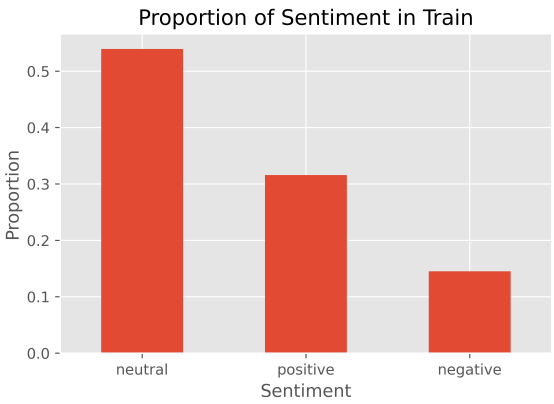

Figure 1: Sentiment Distribution

We follow the methods used in non-deep-learning models to explore text data. The following processings on texts are performed:

- convert to lower case
- remove digits
- remove standard stopwords
- use stemmer to unify tenses of words. We picked the *PorterStemmer* for scalability.

Plotting the popularity of unigram, bigram, and trigram, we observe not much semantic differences are there across different n-grams, but the counts are different, suggesting that incorporating all n-gram features may bring benefits to the final performances.

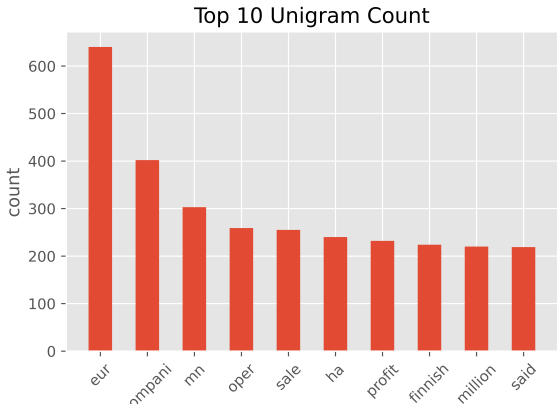

Figure 2: Unigram Count

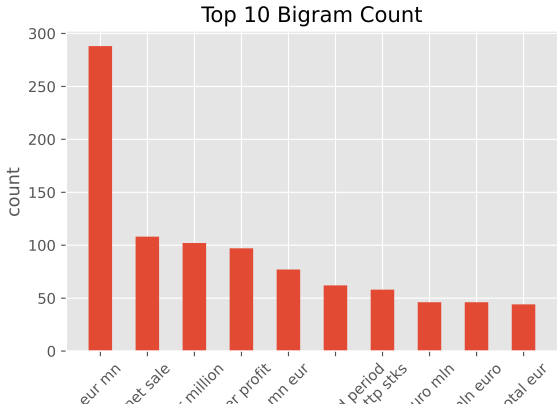

Figure 3: Bigram Count

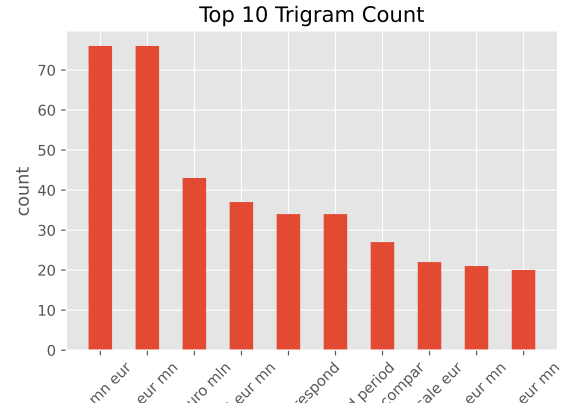

Figure 4: Trigram Count

We also extract sentiment scores using the pretrained model available from the nltk package. This methods decompose a sentence into 4 types of emotions: *pos*, *neg*, *neu*, *compound*.

Notice that non of the single metrics separate three labels well. For instance, neg intensity could separate negative sentiments from others, while neu and pos sentiment indtensity could only separete neutral from positive and negative. This suggests that using all four features is necessary in reaching a good performance in the final outcome.

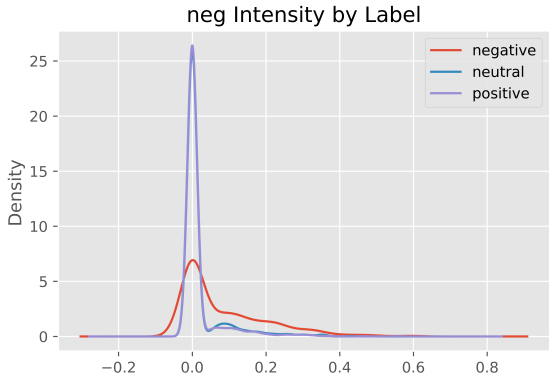

Figure 5: neg Sentiment Intensity

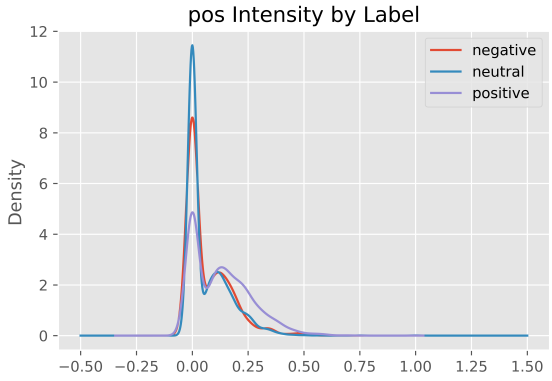

Figure 6: pos Sentiment Intensity

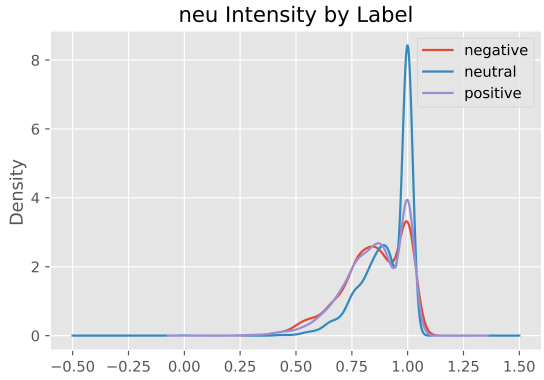

Figure 7: neu Sentiment Intensity

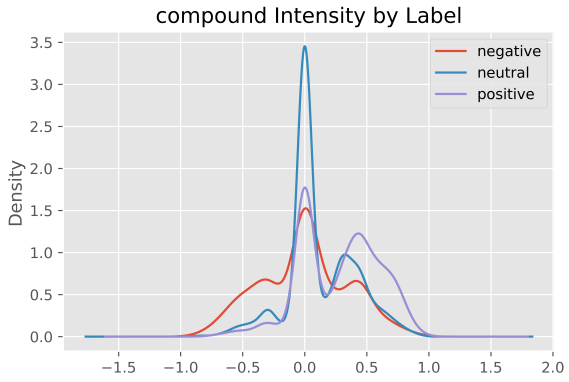

Figure 8: compound Sentiment Intensity

3 Model Building

To have a comprehensive cover of the mainstream machine learning models, we test on the followings:

Table 2: Model Choices

Model Type	Model Name
Linear	Logistic
Bagging	Random Forest
Boosting	XGBoost, LightGBM
Neutral Network	LSTM
Transformer	BERT

As foreshadowed above, we will use the following features for non deep learning models, after converting to lower cases, removing stopwords, using stemming, and remove digits:

- Sentiment Intensity Scores for all four sentiments
- Top 2000 Unigram Counts
- Top 2000 Uni-bigram Counts
- Top 1000 Bigram Counts
- Top 500 Trigram Counts

For deep learning models, we convert each words to a sequence of index, where index is the top 2000 popular words in the train dataset. We pad each sentence to a max length of 40.

4 Model Testing

To evaluate each model, we shuffled the train dataset, and split it into training and validation splits, with a 80/20% ratio and data is split in a stratified fashion. We ran each model for 10 differnt splitting and report the averaged accuracy and F1 scores. Each model is tuned to its best performing parameters. We found out that **BERT model performed the best across all models**. The following is the reported average metrics:

Table 3: Training Acc and F1 (Weighted)

Name	Train accuracy	Train F1
Logistic	0.937	0.936
Random Forest	**0.945**	**0.944**
XGBoost	0.86	0.856
LightGBM	0.851	0.848
LSTM	0.751	0.747
BERT	0.939	0.939

Table 4: Validation Acc and F1 (Weighted)

Name	Val accuracy	Val F1
Logistic	0.67	0.664
Random Forest	0.671	0.658
XGBoost	0.692	0.683
LightGBM	0.678	0.67
LSTM	0.632	0.623
BERT	**0.755**	**0.754**

Notice that bagging model tends to overfit, observed by that the Randomforest model has the highest training accuracy and f1 score. It no longer has any differences when testing in the validation set.

An additional plot is made for comparing across different train test split: we plot the model performance on validation acc vs f1 at each train test split. This strike lead in BERT further convinces the superiority of transformer-based models in sentiment analysis.

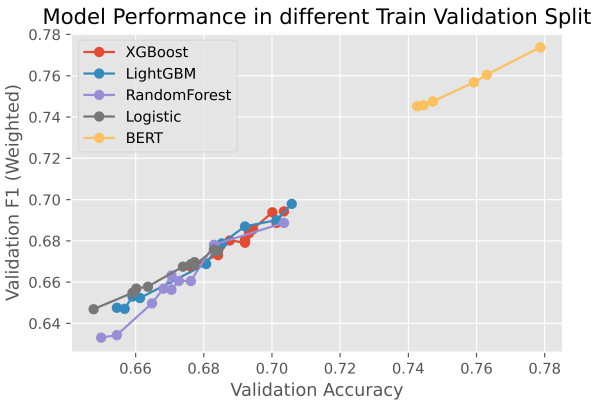

Figure 9: Model Performance in each train val split

We report our final prediction using the BERT model.

5 Potential Improvement

Stratification in train test split is one way to ameliorate the impact of class imbalance. We also propose using a custom-weighted accuracy to account for misclassifying positive and negative sentiments.

This is supported by the following domain knowledge: it is relatively cheap to misclassify neutral to other classes, since an incorrect trading signal would not bring much impact to the portfolio return; but it is detrimental to fail to catch the trading opportunity when positive and negative sentiments are observed. So we assign higher weights to samples that have positive or negative ones. Also, since it is generally **easier to take long position than short ones**, we assign higher weights for positive than negative sentiments. This results in a custom weighted scheme:

- Positive: 1.5
- Negative: 1
- Neutral: 0.7

This assignment is arbitrary and is subject to further improvements.

We observe the following: in ALL cases, the weighted accuracy is less than the equally weighted accuracy, meaning that the algorithm is easier to identify neutral ones, which is relatively useless in trading. One example in xgboost equal vs weighted accuracices across 10 different train test split is shown in the following:

Table 5: XGBoost Equal vs Weighted Validation Acc

Equal	**Weighted**
0.6921	0.6623
0.7001	0.669
0.6921	0.664
0.6944	0.6615
0.7013	0.6709
0.6933	0.6619
0.6842	0.6548
0.7035	0.6738
0.6876	0.6594
0.6762	0.645

A custom loss function adjustment is necessary in both traditional machine learning models and the deep learning ones.

6 Conclusion

In this project we have empirically demonstrated that BERT model is the best performing one in sentiment analysis on financial text data. Various feature engineering techniques are employed to boost performances of traditional machine learning models, but BERT model still tops the list.

Tying back to financial domain knowledge, we propose a further adjustment scheme based on the sentiment labels, since we would like to capture all trading opportunities and are relatively ignorant of the neutral samples. A asymmetric custom loss function should be used for more practical purposed to help further improve returns for quant investors.