Comp 4332 Project 1

Wu Chi hsuan (20657213), Wu Tai Ling (20561658), Lee Hao Yu (20656013)

Introduction

In this project, we trained several machine learning pipelines to do sentiment analysis and classification on the text and compared their performances. Our report contains the procedure of feature extraction, model selection, experiment result, analysis, and conclusion.

Feature Extraction

Text transformation

For the review text transformation, we applied the Tf-Idf vectorization as well as the hyperparameter tuning as follows:

- **Tf-Idf Vectorizer**: We compared the performance between Tf-Idf vectorizer and count vectorizer in the sklearn package. We found that Tf-Idf vector performed better.
- **N-grams**: We experimented with different N-gram ranges and found that the combination of unigram and bi-gram (i.e. ngram_range=(1, 2)) is the best.
- **Stopwords**: Since the default stopwords set contains some words as potential candidates for negative words, we decided not to filter the stop words.

Numerical Data

On top of the existing numerical data columns (cool, funny, useful), we tried to extract some other features as follows:

- **Word count in the text**: It has a roughly decreasing trend with respect to stars. Higher ratings roughly refer to shorter reviews and the visualization plots are attached in the appendix.
- **Sentiment scores** (pos, neu, neg, compound): The sentiment scores are collected through the SentimentIntensityAnalyzer object in the nltk package. The correlation between positive scores and stars is about 0.5 while that between negative scores and stars is about -0.5. The correlation plots are attached in the appendix.

Methodology and Experiment Result

The whole dataset is preprocessed by ColumnTransformer() according to the data type:

- Text data: Tf-Idf Vectorization
- Numerical data (cool, funny, useful, nltk sentiment, num_of_words): Standardizing
- Irrelevant data (business_id, review_id, user_id): Drop it

The column transformer provided us the flexibility of deciding whether we should feed the sole text or whole analytical data (text + numerical data) into the classifier.

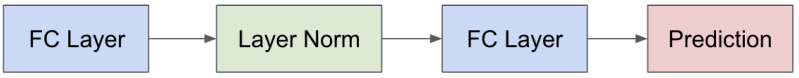
Joint Vector

We concatenated the Tf-Idf vector and the vector consisting of numerical data. We compare the performance of using joint vectors to using text data solely.

MLP Classifier

We selected the features by setting minimum frequency and maximum frequency thresholds

to improve model performance. The designed model is as follows. The purpose is to examine whether increasing model size and using Layer Norm can improve performance.



NNRank
As suggested by [1], we mapped the labels as follows:

1 : [1,0,0,0,0], 2 : [1,1,0,0,0], 3 : [1,1,1,0,0], 4 : [1,1,1,1,0], 5 : [1,1,1,1,1]

The purpose of doing so is to incorporate the ordinal relationship of the rating by using vector representation and avoid treating the rating as unrelated individual classes. The label is assigned to each data by checking the distance between the predicted vector and the ground truth vectors.
[1] (Jianlin Cheng, A neural network approach to ordinal regression (2008), IEEE International Joint Conference on Neural Networks)

Model Selection	Macro-f1	Accuracy	Model	Macro-f1	Accuracy
Logistic Regression	0.54	0.632	MLP Classifier	0.56	0.648
MLP + NNRank	0.58	0.651	MLP + Joint Vector + NNRank	0.57	0.652

From above, we can see that adding additional FC layers and layer norm can improve macro-f1. The performance can further improve after using NNRank. Lastly, we combine NNRank, Numerical data, and our designed MLP module and reach 65.2% accuracy.

Analysis

We analyzed the learned weight of the logistic regression model. For the weight corresponding to the 5-star probability prediction, the entries with larger values match the words that could lead to better ratings. By contrast the entries with lower values match the words that could lead to worse ratings. The visualization of positive and negative word clouds is in the appendix. Some good examples are found below:

- Words match with positive ratings: cheap, dessert, authentic, fairly, pleasant, fast…
- Words match with negative rating: worst, horrible, hard, disappointing, terrible…

Conclusion

In conclusion, we systematically explored that the following transformations and feature engineering techniques were helpful for performing sentiment analysis on Yelp:

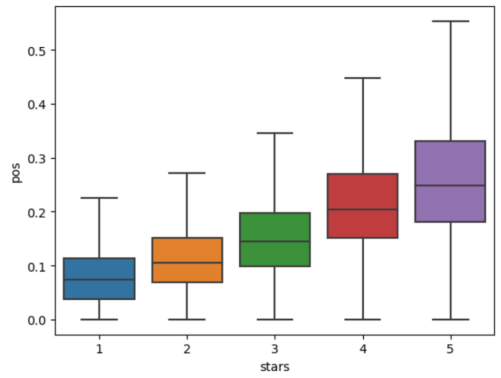
- Numerical data is informative to sentiment analysis
- From the result of NNRank, we show incorporating ordinal relationships is important
- Layer norm and increasing model size can improve the performance

Appendix

The correlation plot between the result from nltk sentiment analyzer and the stars. (From left to right, top to bottom is pos, neg, neu, compound. In each plot, the order of the star is from 1 to 5)

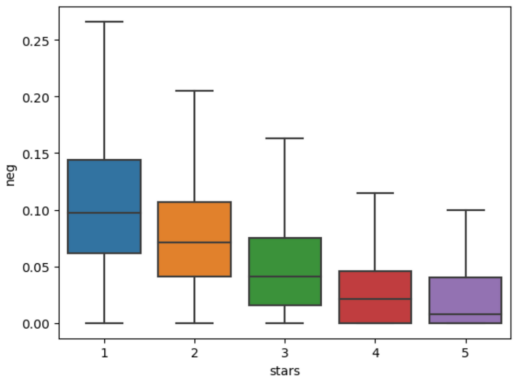
```
sns.boxplot(data=train_df, x='stars', y='pos', showfliers = False)
```

```
<AxesSubplot: xlabel='stars', ylabel='pos'>
```



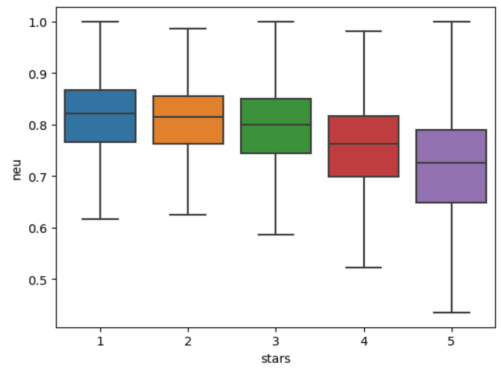

```
sns.boxplot(data=train_df, x='stars', y='neg', showfliers = False)
```

```
<AxesSubplot: xlabel='stars', ylabel='neg'>
```



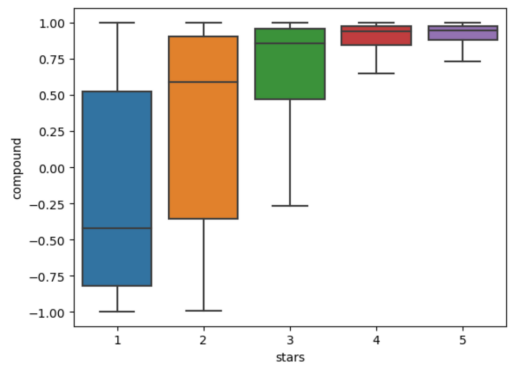

```
sns.boxplot(data=train_df, x='stars', y='neu', showfliers = False)
```

```
<AxesSubplot: xlabel='stars', ylabel='neu'>
```




```
sns.boxplot(data=train_df, x='stars', y='compound', showfliers = False)
```

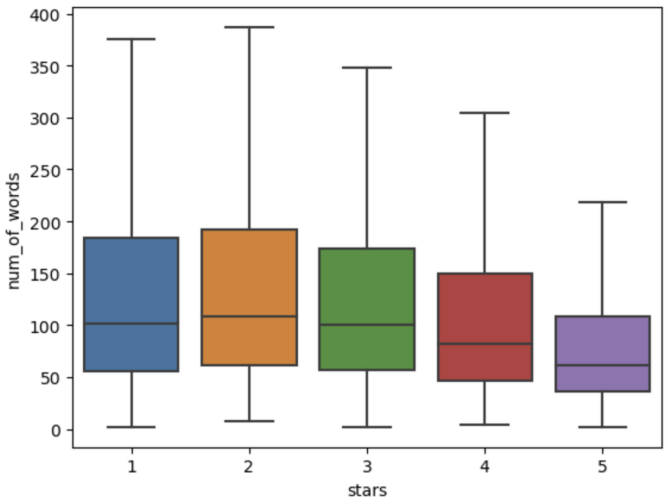
```
<AxesSubplot: xlabel='stars', ylabel='compound'>
```



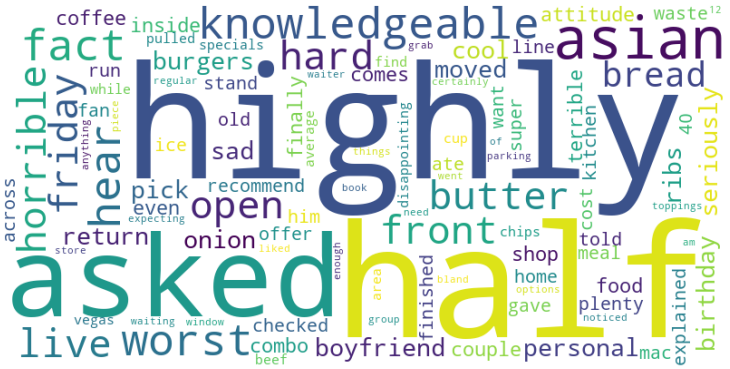
The correlation plot between the review word count and the stars.

```
# Distribution of 'text word count' w.r.t. stars
sns.boxplot(data=train_df, x='stars', y='num_of_words', showfliers = False)
# We can see an approximately decreasing trend of word count w.r.t. stars
```

```
<AxesSubplot: xlabel='stars', ylabel='num_of_words'>
```



The positive (top) and negative (bottom) word cloud indicated by logistic regression weight.



The best validation scores we got with TFIDF text vectorization, standardized numerical data, MLPRegressor as final classifier with NNRank transformed labels

	precision	recall	f1-score	support
1	0.78	0.71	0.74	292
2	0.38	0.36	0.37	163
3	0.42	0.41	0.41	232
4	0.47	0.53	0.50	421
5	0.82	0.81	0.81	892
accuracy			0.65	2000
macro avg	0.57	0.56	0.57	2000
weighted avg	0.66	0.65	0.65	2000

```
[[207  53  13  12   7]
 [ 43  58  54   7   1]
 [ 11  30  94  82  15]
 [  3   6  49 224 139]
 [  3   6  13 149 721]]
accuracy 0.652
```