COMP4332 Project 1: Sentiment Analysis

DO Van Quyet, TRINH Van Hoan, NGUYEN Kha Nhat Long
The Hong Kong University of Science and Technology
{vqdo, vhtrinh, knlnguyen}@connect.ust.hk

1. Data Exploration

The data used in this project is from the Yelp Dataset. The data consists of reviews of different places by different users. The goal of this project is to build a model to predict the number of stars that a customers will give in a review.

All data analyses are based on the training data set.

1.1. Features analysis

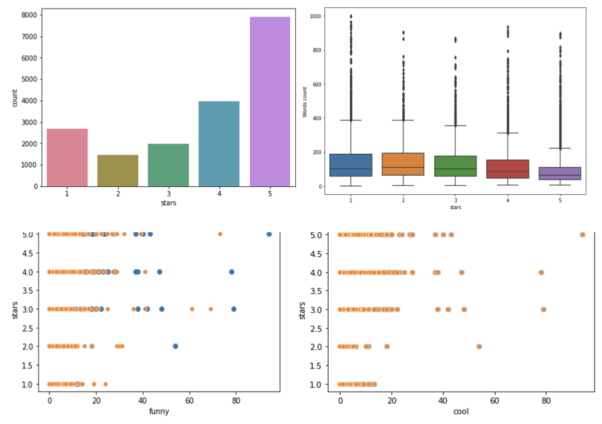

Figure 1: Top Left: histogram of stars, Top Right: boxplot of review's length, Bottom left : Scatter plot of funny and stars, Bottom right: Scatter plot of cool and stars

The data is highly skewed towards 4 and 5 stars. The data for 2 stars is especially small compared to other classes. For each class, the length of text distributions are almost similar.

Other features including 'cool', 'funny', 'useful' do not have a significant relationship with 'stars' (as expected from the real world).

Thus, we focus on 'texts'(the reviews), to predict 'stars'.

1.2. Texts exploration

Using the list of positive/negative words from Bing Liu's Opinion Lexicon, we examine the positive/negative words counts for each class.

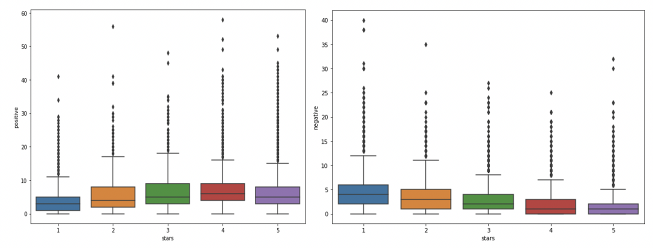

Figure 2: Positive(left) and Negative(right) words counts.

Figure 2 shows that the negative words tend to decrease from 1 to 5 stars and the positive words tend to increase from 1 to 5 stars. However, the positiveness/negativeness of reviews for 4 and 5 stars are mixed together. Also, the differences of sentiment word counts between classes are observable yet inadequate to differentiate reviews of different classes.

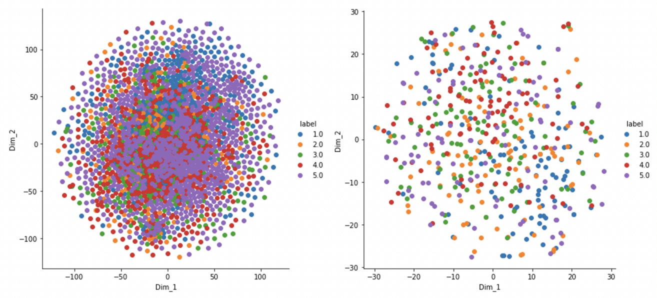

Figure 3: TSNE plot of IF-IDF vector representation of reviews in train dataset. Left: full set of data, Right: a smaller subset.

In Figure 3, focusing on the cluster, we can see that the clusters might mix the (5-star and 4-star) and (1-star and 2-star), which might confuse the models. That suggests the IF-IDF vectorizer is not sufficient to represent reviews.

2. Experiments

To find the model that captures the sentiments of the reviews, first we followed the two given architectures, Statistical Machine Learning with TF-IDF+Logistic Regression and Deep Learning with RNN+Word2Vec, as two baseline models. Experiments with different setups and parameters showed that these two approaches can achieve decent results. We further employ Tranformer-based State-of-the-art classification models.

Since we focused on macro-f1, we compared and selected the best model according to this metric. Noticing the sensitivity of macro-f1, especially in a skewed dataset, we also took model accuracy as a reference.

All experimental results can be found in Table 1.

2.1. TF-IDF Vectorization and Logistic Regression

We started with a simple baseline, as combination of two classical tools for text classification, TF-IDF Vectorization and Logistic Regression (LR). TF-IDF vectorizer characterize each text sequence by the 'significance' vector based on words of the sequence. This gives the input of the classification LR model.

We alternatively and manually searched the best hyperparameters for each component. First, we fixed the vectorizer and find the best regularization coefficient and class weight of the classifier, then vice versa fixed the classifier and find the best vectorizer's hyperparameters such as ngram and max_features. Finally, the process end up with TF-IDF(ngram=(1,4), max_features=30000, lowercase=False) and LR(C=3, class_weight=[3,6,4,2,1]), and shows a significant improvement.

At the end, from the observation on the training data that samples from class 2,3,4 are challenging to be differentiate, and as heuristics from many previous works on Sentiment Analysis, we intuitively propose and experiment an ensemble (135+12+45) of three LR models. In this ensemble, one LR model will treat samples from class 1 and 2 as NEGATIVE, 3 as NEUTRAL, 4 and 5 as POSITIVE, then learn to classify text into these three categories. The other models focuses on classfying samples from class 1 versus 2 and 4 versus 5. Besides, we further examined a similar ensemble (135+234), where we first treated samples from class 2,3,4 as NEUTRAL, 1 as NEGATIVE, and 5 as POSITIVE. We also perform hyperparameter tuning, and get a slightly better result (Table 1).

2.2. Word Embeddings and RNN

Standard operations to convert texts to stemmed tokens were firstly processed. Simple deep learning architectures, vanilla CNN and RNN architectures were firstly examined, and preliminary results show that RNN is more stable and more accurate.

To tackle the disadvantage of vanilla RNN, we considered LSTM, GRU, and Bidirectional LSTM layers to gain more control over the memory of the model. Different setups and parameters show that 3 directions yielded similar results.

To further improve the model, Word2Vec was used to achieve dimensionality reduction and capture contextual similarity. For this, FastText and GloVe were considered, 2 pre-trained word embeddings. Through experimentation with different setups and parameters, FastText resulted in better accuracy and macro-f1 score than GloVe. A good setup for this approach is FastText, 1 GRU layer, and 2 MLP layers.

This baseline model yields decent results but it can't surpass the previous model which is less complex.

2.3. Transformer-based model

We applied several classification models of the Transformer family. Those models, also called Language models, are powerful tools to embed various information of textual data. With the attention mechanism, their capacity to encode the tree structure of textual data is extended, thus they give more meaningful and comprehensive vector representations of words and sentences. Also, the fact that they are pretrained on huge corpora, such as Book Corpus and English Wikipedia, makes their vector representation robust for textual data from different domains, hence frequently being used for down-stream tasks such as Sentiment Analysis, (Extractive) Summarization, Named-Entity Recognition, etc.

We finetuned the following pretrained language models with an additional classification head on our provided Yelp Sentiment Analysis dataset.

BERT is a bi-directional transformer for pre-training over a lot of unlabeled textual data on self-supervised training tasks Masked Language Model (MLM) and Next Structure Prediction (NSP) to learn a language representation that can be used to fine-tune for specific machine learning tasks. Here, the base-cased and base-uncased versions of BERT are experimented, with default hyperparameters.

RoBERTa is retraining of BERT training methodology, 1000% more data and compute power. In fact, it removes NSP, introduces dynamic masking for MLM, and importantly uses 160GB of text for pretraining. As our preliminary result shows the higher performance of RoBERTa than BERT, hyperparameter-tuning for RoBERTa is conducted with the aim to boost the model performance. In specific, we experiment:

- RoBERTa-base with different dropout p=0.3: Empirical modification,

- RoBERTa-base with different maximum lengths for input sequences max_len=512: By histogram of sequence length, there is a proportion of data having lengths from 256 to 512. Chunking the sequence to a maximum length of 256 may lose some meaningful information, and
- RoBERTa-base uncased variant: Capitalized words in fact effectively represent the sentiment of the reviewers. Since RoBERTa does not have an uncased version, i.e RoBERTa is pretrained on cased corpora, we uncased all provided data before feeding it to the model.
- RoBERTa-large: To examine the effectiveness of model size. Unfortunately, the training process has unexpected issues, thus we do not get the result for the model.

Moreover, inspired by the research on finetuning procedure, we adopt the proposed fine-tuning strategy to RoBERTa-base. In the first few steps, we fix the weights of the stem RoBERTa and only train the head classifier (called Linear Probing), then finetune the whole model.

Training: Data preprocessing is not necessary, since those models are robust in processing raw data. We train models 3 epochs with batch size 32 and learning rate 1e-5, with maximum length for input sequence as 256, dropout 0.4 as default hyperparameters. Some hyperparameters are changed according to the nature of the dataset in a fix-all-except-one manner, for comparison and best-model selection purposes. All experiments are run once due to the lack of time and resources.

2.4. Results and Analysis

We presented results on two focused metrics, macro average F1 score, and accuracy, of all examined models in Table 1. First, we can observe that the Transformer-based model outperforms all previously discussed models. The significantly large gap in both metrics indicates the higher capacity for text representation and the advantage of pretraining on large common-text datasets of those language models, aligning with many previous works. In addition, the gap between performances of BERT and RoBERTa is intuitive due to the same reason, as RoBERTa is robustly optimized from BERT with a larger size training set and other mechanisms.

Second, we analyzed results from variants of RoBERTa. The uncased version performed worse than others, likely due to the contribution of case sensitivity to sentiment. However, the similar performance of BERT-base-uncased and -cased suggested that the decrement in performance of our uncased RoBERTa originates from the pretraining procedure of RoBERTa. Other hyperparameter changes did not

Models	Macro-F1	Accuracy
TF-IDF(), LR(C=1)	0.5279	
* replaced LR-best	0.5465	
* replaced TF-IDF-best	0.5627	
Ensemble (135+234)	0.5654	0.6530
Ensemble (135+12+45)	0.5763	0.657
BiLSTM	0.4357	0.6085
GRU	0.4777	0.5735
GRU+GloVe	0.5291	0.6400
GRU+FastText	0.5761	0.6550
RoBERTa-base		
* default	0.6584	0.6970
* dropout=0.3	0.6540	0.7060
* uncased	0.6478	0.6950
* max_len=512	0.6576	0.7035
* LP-FT	**0.6648**	**0.7060**
BERT-base		
* cased	0.6192	0.6615
* uncased	0.6216	0.6710

Table 1: Experimental result: macro-f1 and accuracy of all examined models on validation dataset.

yield a significantly different result. Nonetheless, the LP-FT mechanism expectedly improved the model, supporting the argument that the machine helps to prevent learned features of the pretrained RoBERTa model from being distorted by the first few learning iterations.

In addition to model-centric analysis, we made some comments on the given dataset. The classification report on the validation set of the best model (Table 2) shows that it is harder to correctly classify samples from classes of 2,3,4 stars. We argued that the difficulty comes from the nature of the data. Samples from classes 2,3,4 tend to be neutral, or in other words, in a lower degree of sentiment than samples from classes 1 and 5. That makes the differentiation of those samples between each other and samples from classes 1 and 5 not trivial. Besides, from our close observation of the data, we suggest that the text-star relation is not consistent, especially in class 2,3,4, hence making the prediction harder.

3. Conclusion

In this project, we explored a part of the Yelp Review dataset and acknowledged the complication of predicting stars for reviews. We employed a wide range of models, from a statistical model as a combination of TF-IDF and Logistic Regression, to recent SOTA Deep Neural Networks, in specific, Transformer-base models, in order to perform Sentiment Analysis on the Yelp Review dataset. Overall,

class (stars)	precision	recall	f1-score	support
1	0.8399	0.8369	0.8384	282
2	0.4939	0.5956	0.5400	136
3	0.5888	0.5472	0.5672	212
4	0.5347	0.6288	0.5779	466
5	0.8469	0.7588	0.8005	904
accuracy			0.7060	2000
macro avg	0.6608	0.6734	0.6648	2000
weighted avg	0.7218	0.7060	0.7115	2000

Table 2: Classification report of RoBERTa LP-FT variant on validation set.

with the advantage of being pretrained on huge common-text corpora, all Transformer-based models yielded higher scores than previously invented models. Among those, one with the Linear Probing-then-Finetuning mechanism gave the best performance. We used the model to generate a prediction for test data as the **submission**. Code to reproduce results for transformer-based models is available here.