Predicting the Star Rating of Yelp Reviews

Silvan Kübler, Nicolas Peyer

October 27, 2023

Abstract

This study aims to predict Yelp star ratings from textual reviews using a multi-class classification approach. Five supervised algorithms were employed on a preprocessed Yelp dataset, using both custom and scikit-learn pipelines. The models were evaluated based on the metrics accuracy and one-off accuracy and the highest scoring model was further evaluated. Distinguishing between 2-4 star ratings proved challenging. Adversarial testing further revealed the model's limitations with regards to complex reviews. Future work will explore larger datasets for better results.[1]

[1]Github repository containing the source code: https://github.com/Sirofjelly/nlp-multiclass-classification

1 Introduction

In our project we worked on a multi-class classification problem. Our goal was to predict the star rating of Yelp reviews based on their textual counterparts. Predictions of this sort have many real-world applications, however, particularly in the context of customer insights. Efficient review processing allows businesses to improve their service quality and as a consequence their customer satisfaction.
To determine the best strategy to solve this problem we used five different supervised learning algorithms and compared their performance. A similar project was already done by [1], however, instead of only comparing models, we conducted a thorough examination of one specific model, assessing its robustness against adversarial inputs and variations in the data.

2 Dataset

In our project proposal, we planned on working with the *amazon_us_reviews*[2] dataset, however, it was removed and therefore we had to chose another comparable dataset. As a substitution we decided to use the *yelp_reviews_full*[3] dataset from huggingface. This English, industry unspecific review dataset was extracted from the Yelp Dataset Challenge 2015 and first used in the text-classification paper "Character-level Convolutional Networks for Text Classification" [4]. It consists of 700'000 data points, which have the following structure:

```
{
    'label': 0,
    'text': 'I got \'new\' tires from them and within two weeks got
        a flat...'
}
```

Each 'label' entry can take a value between 0 and 4, mapping to corresponding review ratings of 1 to 5, each 'text' entry contains the textual review that determines the value of 'label'. When downloading the dataset the samples are randomly split into a training set of size 650'000 and a testing set of size 50'000. The classes are evenly distributed which means that each label contains the same number of samples after data splitting.

3 Preprocessing

Our text preprocessing approach consisted of two steps. We first processed the input data using our own text processing pipeline. This pipeline consists of a normalization function that converts the text data to all lowercase. In another processing step it removes stopwords and punctuations and finally, it lemmatizes the data.
To implement the normalization function, we used standard python list comprehensions. This normalization step ensures a consistent data treatment. For example, the words 'ME', 'Me', 'mE', and 'me' are all treated as one feature as a consequence, whereas without this normalization step, each of these words would become a single feature.
We removed stopwords such as 'the', 'and' or 'in' from the text data because they hold little semantic meaning. To determine a list of stopwords that should be removed, we used *stopwords* for the English corpus from the nltk library[4].
Since punctuations similarly add little semantic meaning and we removed them as well from the text data, using the punctuation's defined in *string.punctuation*.

[2] https://huggingface.co/datasets/amazon_us_reviews
[3] https://huggingface.co/datasets/yelp_review_full
[4] https://www.nltk.org/search.html?q=stopwords

Lastly, we transformed the words to their lemma to reduce the feature space and to ensure consistent data treatment. We achieved this conversion by using the *WordNetLemmatizer()* from the nltk library[5].
The medium article greatly inspired our second text preprocessing step [2]. In our data exploration phase we also used wordclouds to determine the most frequently occurring words in our dataset. We similarly identified that the most common words were descriptive or neutral words. Furthermore, we noticed that many words that occurred frequently in texts with labels 0 also occurred frequently in texts with label 4. As a result the distinction between these classes would be difficult without further processing steps. To address

Figure 1: 1-Star Word Cloud

Figure 2: 5-Star Word Cloud

this problem we used the sklearn function *TfidfVectorizer()*[6]. This function vectorizes the textual input data by creating a TF-IDF matrix. Each matrix entry represents a specific word (feature) and sample, where the TF-IDF score is stored. The more often a word is present in a dataset, the lower it's TF-IDF score, because the score is calculated by TF * IDF. Since the IDF score tends towards 0 if a specific feature is contained in many samples, and only the 700 highest scoring features are considered in our models, this vectorizer is very suitable for our problem.
We also integrated another sklearn function, *StandardScaler()*[7], into the *text_preprocessing_pipeline*[8] from sklearn. This function simply standardises the vectorized features to have a mean of 0 and a standard deviation of 1. The features were scaled to ensure a more meaningful comparison between features.

4 Model training

Following our project proposal, we started off training three different supervised learning algorithms: Naive Bayes (NB)[9], Support Vector Machine (SVM)[10] and Multi Layer Perceptron (MLP)[11]. We chose to implement these models since they are suitable for multi-class classification and represent three different model categories each of increasing complexity. Having models of varying complexity allows for an interesting analysis of the models' performance. Simpler models often offer great computational efficiency and inter-

[5] https://www.nltk.org/_modules/nltk/stem/wordnet.html
[6] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
[7] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
[8] https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
[9] https://scikit-learn.org/stable/modules/naive_bayes.html
[10] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[11] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

pretability, but they can lack performance. On the other hand, more complex models often suffer with regards to interpretability and computational efficiency but perform better.
NB is a statistical model and the simplest model in our project. It assumes feature independence which, however, is often not the case with text data. SVM is its own model category and is able to deal efficiently with high dimensional spaces and non-linear decision boundaries thanks to the kernel trick. MLP belongs to the category of neural networks and is the simplest version of such a network. In spite of this, it can easily model complex non-linear relationships because of it's adaptability to a specific task. All three models were implemented using scikit-learn.
Since the performance of the NB and the MLP models was underwhelming, we decided to implement another, more sophisticated statistical model and another, more complex neural network. As the statistical model we chose logistic regression. This model often serves as a baseline and performs well, if the relationships are of linear nature. We implemented the logistic regression (LR) also using scikit-learn. For our more complex neural network, we chose to implement a deep neural net (DNN) using skorch[12]. In this model we used a dropout layer to prevent overfitting and implemented two hidden layers in hopes of increasing the performance of the model.
Each of the models was trained with the same amount of preprocessed data, i.e., 20'000 samples. If a model performed exceptionally poorly or well, we conducted a grid search in order to optimize it's performance further. This was the case for the LR model and the DNN.

5 Evaluation

As can be seen in Figure 2 our training data was distributed evenly and consequently a performance comparison of the models via accuracy score is permissible. Another reason, why we chose accuracy as performance measure was that the other performance measures precision and recall, are ('only') relevant if the costs of false positives or false negative respectively are high, which is not the case in our project.

	Accuracy	One-off Accuracy
NB	45.60%	83.53%
LR	50.62%	87.40%
Optimized LR	52.80%	89.45%
Robustness LR	52.02%	88.40%
SVM	51.10%	87.42%
MLP	49.70%	86.70%
NN	21.75%	41.80%
Optimized NN	49.95%	88.45%

Table 1: Performance Comparison

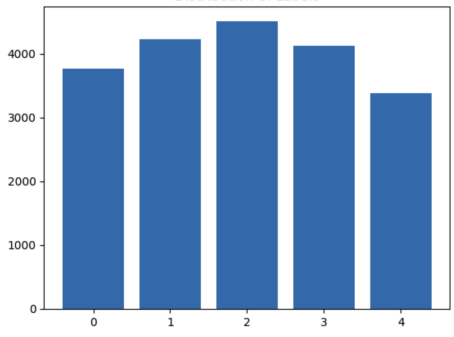

Table 2: Distribution of training data labels

The first column of Table 1 illustrates the overall performance of each model. It can be seen that the optimized LR model, the SVM model and the optimized NN performed the best, where the optimized LR model had the highest accuracy score of 52.80%. A similar picture can be observed when looking at the second column 'One-off Accuracy', whereby the optimized LR model again shows the best performance with a one-off accuracy of 89.45%. This accuracy measurement was introduced to take the subjective nature of reviews into account and extends the accuracy's definition of 'correct prediction'. Label predictions that are off by 1 i.e., 1 class higher or lower than the true label, are also considered to be correctly predicted.

[12]https://skorch.readthedocs.io/en/stable/index.html

Since the optimized LR model was unbeaten with regards to performance and computational efficiency, we decided to analyse it in more depth. Therefore, we created a feature importance table (see Figure 3) where we extracted the top ten most important features per label. For example, for the classification of a text as label 0 the feature "worst" had the highest impact with a weight of 0.417. Furthermore, it can be seen that there is some feature overlap per label, suggesting that a clear distinction between all labels is not possible.

y=0 top features		**y=1** top features		**y=2** top features		**y=3** top features		**y=4** top features	
Weight?	**Feature**	**Weight?**	**Feature**	**Weight?**	**Feature**	**Weight?**	**Feature**	**Weight?**	**Feature**
+0.417	worst	+0.457	<BIAS>	+0.721	<BIAS>	+0.346	great	+0.428	great
+0.303	horrible	+0.216	bland	+0.188	ok	+0.264	<BIAS>	+0.326	best
... 338 more positive ...		+0.203	worst	+0.177	decent	+0.249	delicious	+0.287	delicious
... 353 more negative ...		+0.191	ok	+0.171	good	+0.200	love	+0.287	love
-0.306	best	+0.187	terrible	+0.151	bit	+0.182	excellent	+0.284	amazing
-0.314	amazing	+0.183	mediocre	+0.135	average	+0.168	good	*... 340 more positive ...*	
-0.317	love	*... 364 more positive ...*		+0.124	okay	*... 357 more positive ...*		*... 351 more negative ...*	
-0.318	delicious	*... 327 more negative ...*		+0.116	pretty	*... 334 more negative ...*		-0.275	decent
-0.343	excellent	-0.168	love	*... 372 more positive ...*		-0.155	bland	-0.323	bland
-0.345	friendly	-0.176	perfect	*... 319 more negative ...*		-0.218	horrible	-0.365	worst
-0.583	great	-0.223	great	-0.121	extremely	-0.255	worst	-0.387	rude
-0.836	<BIAS>	-0.267	delicious	-0.139	even	-0.285	terrible	-0.606	<BIAS>

Figure 3: Feature Importance Table

This assumption is supported by the low accuracy score of the model. The ambiguity regarding classification further becomes evident when looking at the confusion matrix of the optimized LR model. While many instances of labels 0 and 4 were correctly classified, the model struggled to distinguish texts with labels 1 to 3. In these cases the model classified many instances in the one-off range of the correct prediction.

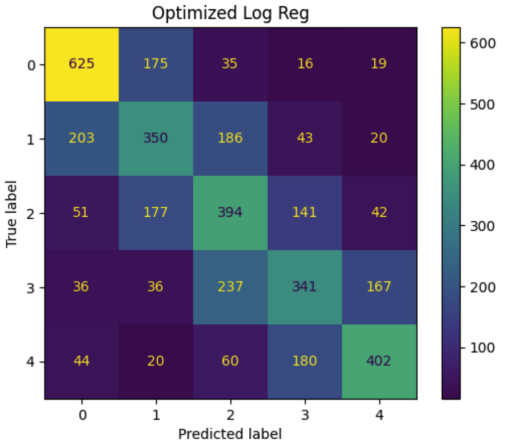

Figure 4: Confusion Matrix of the LR

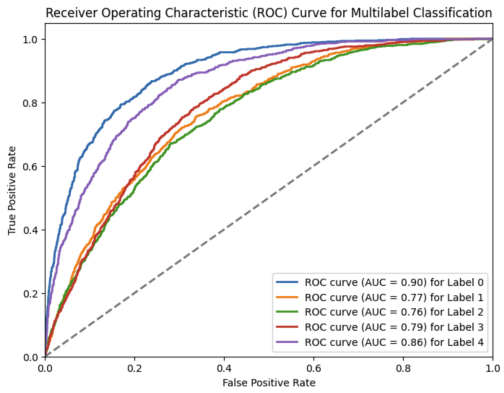

Figure 5: ROC Curve of the LR

Figure 5 displays the area under the Receiver Operating Characteristic (ROC) curve. It is a measure for the classifier's ability to distinguish between positive and negative classes of a label and provides further evidence that the model struggles to predict the middle labels. As can be observed the labels 0 and 4 show the highest area under the curve (AUC), while the labels 1 to 3 all have lower values indicating reduced distinguishability.
Lastly, we assessed the robustness of the optimized LR model using an adversarial attack and variations in data. For the adversarial attack, we generated 10 deceptively positive (DP) (but actually negative) and 10 deceptively negative (DN) (but actually positive) samples. We then classified these samples and calculated their mean to determine whether the model was able to correctly predict the underlying positive or negative sentiment. Based on the means of 1.7 (DP) and 1.4 (DN) we concluded that the model struggles to correctly predict the underlying sentiment of reviews that require real-world knowledge. To test the

robustness of the model on variations in the data, we extracted the top 50 features from the model and replaced these words in the testing set with corresponding synonyms, to examine whether this would result in a significant performance drop, however, as we can see in Table 1 this was not the case.

6 Future work

The next logical is to train the models on a larger dataset, preferably on the whole dataset to see whether further performance improvements can be made. Unfortunately this was not possible using our restricted hardware. To address the issue of the ambiguous label classification, more advanced text processing techniques like BERT could be used for the embeddings. These have shown to improve NLP tasks significantly [3]. To improve the accuracy further, more advanced model architectures such as recurrent neural networks or transformer based models could be explored. Lastly, a comparison between the accuracy of a human and a machine based classification would be an interesting topic to focus on.

References

[1] Nabiha Asghar. *Yelp Dataset Challenge: Review Rating Prediction*. May 17, 2016. arXiv: 1605.05362[cs]. URL: http://arxiv.org/abs/1605.05362 (visited on 10/26/2023).

[2] Joshua Phuong Le. *Rating Prediction from Review Text with Regularization — Linear Regression vs Logistic Regression*. MITB For All. Sept. 29, 2023. URL: https://medium.com/mitb-for-all/rating-prediction-from-review-text-with-regularization-linear-regression-vs-logistic-regression-df0181fe9c07 (visited on 10/26/2023).

[3] Ian Tenney, Dipanjan Das, and Ellie Pavlick. *BERT Rediscovers the Classical NLP Pipeline*. Aug. 9, 2019. arXiv: 1905.05950[cs]. URL: http://arxiv.org/abs/1905.05950 (visited on 10/26/2023).

[4] Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-level Convolutional Networks for Text Classification". In: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper/2015/hash/250cf8b51c773f3f8dc8b4be867a9a02-Abstract.html (visited on 10/26/2023).