NLP Course

Semantic CorpRate

May 2023

Abstract

Link to project code: `https://github.com/kikikita/semantic_corprate`.

1 Introduction

The problem of predicting a company's rating based on text reviews left by customers is an important one for several reasons. First and foremost, a company's rating can significantly impact its success or failure. A high rating can attract new customers and increase sales, while a low rating can deter potential customers and lead to a decline in revenue. Therefore, being able to accurately predict a company's rating can help businesses make necessary changes to increase customer satisfaction and overall success.

In addition, analyzing customer reviews can provide valuable insights into a company's strengths and weaknesses. Businesses can identify opportunities for improvement and decide how to meet the needs of their customers better, by analyzing the specific aspects of a company that customers either praise or criticize.

Furthermore, predicting company ratings from text reviews can also benefit consumers. With so many products and services available, it can be difficult for people to make informed decisions about which companies to do business with. Consumers could make more informed decisions about which companies to support using provided accurate ratings based on customer reviews.

What sets our approach apart from other methods is that we use natural language processing techniques such as tokenization, stopword removal, and stemming to preprocess the text data. We also use word embeddings to capture semantic similarity between words and phrases, which allows us to better understand the meaning behind customer reviews. Additionally, we implement named entity recognition to extract important entities from text data, which can provide valuable insights into the specific aspects of a company that customers either appreciate or dislike.

By combining these techniques, we are able to develop a highly accurate model for predicting company ratings based on text reviews. This allows businesses to make informed decisions about how to improve customer satisfaction and overall success.

1.1 Team

Our team by responsibilities:
Nikita Girman - DS/Backend
Vladislav Mostovik - DS/MLE
Irina Prokofieva - DS/Backend
Arina Gritsai - DS/DA

2 Related Work

Since we are thinking about using the result of our work in a service where the execution time is also quite important, we decided to take into consideration both classic NLP methods and neural networks.

We used a survey by [Kamran Kowsari, 2020] as a general reference to choose text classification algorithms. Here we got the idea to experiment with two approaches which are Count Vectorizer [Jain, 2021] with Linear Regression and Naive Bayes. These two approaches were chosen as baselines, which in fact appear to be quite high.

Here ([Kamran Kowsari, 2020]) we also thought of an idea of using pre-trained models to get word embeddings. We had chosen pre-trained model for english language from SpaCy library [Shiraly, 2023] to get word embeddings. After that we implemented CatBoost Classifier [Baranyuk, 2021] as a model for gradient boosting (general idea of CatBoost [Liudmila Prokhorenkova, 2019]).

The last approach we thought of were sentence transformers. The advantage of such models is that they do not require mandatory text preprocessing. Here we have found two appropriate variants for our objective: classic BERT model [Jacob Devlin, 2019] and its modification called RoBERTa [Yinhan Liu, 2019]. The reason why we decided to check both was to know whether it is possible to obtain competitive results due to a modified model RoBERTa. After getting the word embeddings we used cross-validation to evaluate the performance of the embeddings on the CatBoost Classifier model [Baranyuk, 2021] for RoBERTa and Gradient Boosted Decision Trees [Chawda, 2022] for BERT.

3 Model Description

3.1 BERT

In this section, we describe our main approach - BERT [Jacob Devlin, 2019] [1]. BERT is an acronym for Bidirectional Encoder Representations from Transformers, transformer - based pre-trained model. BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training.

[1] https://arxiv.org/pdf/1810.04805.pdf

3.2 Training

Before feeding word sequences into BERT, 15% part of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence. This approach is called pre-training.

In the BERT training process, the model receives pairs of sentences (in our case is just one sentence and label) as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other 50% a random sentence from the corpus is chosen as the second sentence. The assumption is that the random sentence will be disconnected from the first sentence. This approach is called fine-tuning. Both approaches are presented on Fig. 1.

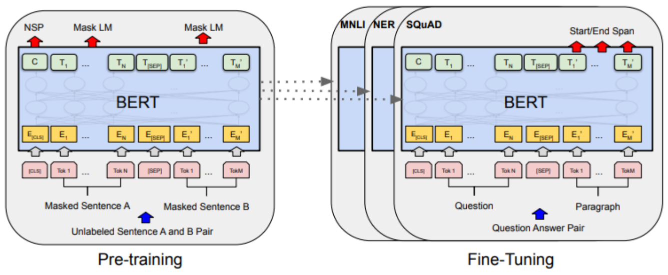


Figure 1: Pre-training and fine-tuning procedures.

3.3 Hyperparameters

We denote the number of layers (i.e., Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A. To fine-tune on our task, we used a BERT-base model (L=12, H=768, A=12, Total Parameters=110M) with batch size of 16 and fine-tune for 3 epochs over the data for all train dataset. We selected default (2e-5) learning rate. Optimizer is AdamW.

3.4 Inference

When we want to make predictions on new test data, then once model training is complete, we can put our goal sentence into BERT, and we will receive the score distribution over label space, where the highest score is prediction of our model for the goal sentence, and the index of this score is label .

4 Dataset

The dataset that we use in our work is taken from a Kaggle competition called "Sentiment Analysis - Company Reviews"[2] handled by [Cortinhas, 2023]. It consists of 100,000 reviews collected from Trustpilot and spans over 45 different companies. The dataset has three columns: Id - Unique id of each review, Review - Customer review (text) and Rating - Number of stars awarded by customer, between 1 and 5. The samples are split according to: 60% train, 10% public leaderboard and 30% private leaderboard. The target is a star rating between 1 and 5. Data distribution shown in the Fig. 2 below (detailed data analysis is presented in our github).

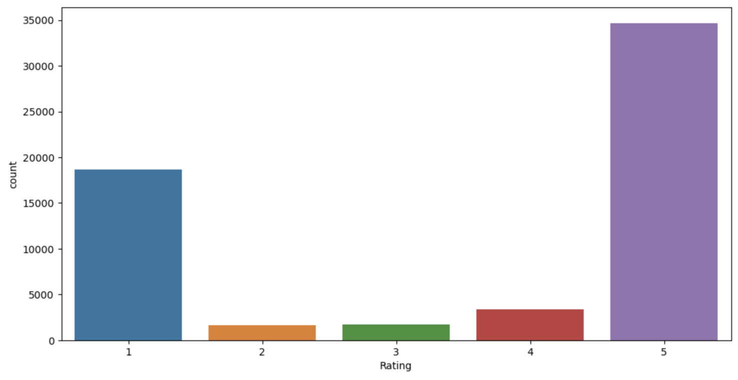


Figure 2: Data distribution.

5 Experiments

5.1 Metrics

When it comes to sentiment classification, the choice of evaluation metric is crucial in determining the accuracy and performance of the model. In this case the target is a star rating between 1 and 5, and because of the natural ordering of the categories a regression metric Mean Absolute Error (MAE) could be used to evaluate predictions.

$$MAE = \frac{1}{D}\sum_{i=1}^{D} |x_i - y_i|$$

where x_i is an observed value for observation i, y_i is a predicted value for the observation i, D is a number of observations.

MAE is a regression metric that measures the average absolute difference between the predicted and actual sentiment scores. The final goal for this case is to build a model that minimizes the MAE on the test set.

[2] Sentiment Analysis - Company Reviews Dataset.

In sentiment analysis, the sentiment score is usually a continuous value ranging from -1 to 1, representing the degree of positivity or negativity in the text.

MAE is preferred over other metrics like Accuracy, Precision, Recall or F1-score, as it considers the magnitude of the errors in the predictions, rather than just their direction. This means that even if the model predicts a sentiment score that is off by a small amount, it will still be penalized for it.

Moreover, MAE is a robust metric that is less sensitive to outliers or extreme values in the data, making it a suitable choice for sentiment classification tasks, where the sentiment scores can vary widely.

In conclusion, the MAE metric provides a reliable and accurate measure of the performance of a sentiment classification model, and is therefore a popular choice in sentiment analysis research.

5.2 Experiment Setup

The exact design and other important details of each experiment is described in the appropriate subsection below.

5.3 Baselines

5.3.1 CountVectorizer and LinearRegression

In this experiment, we aimed to perform sentiment analysis on company reviews using the Count Vectorizer and Linear Regression techniques. The goal was to predict the sentiment of the reviews based on the words used in them. We used the Mean Absolute Error (MAE) metric to evaluate the performance of the model.

The first step in our experiment was to preprocess dataset by removing stop words, punctuations, digits, leaving only words consisting of ASCII-characters, lemmatizing them and lowercasing the text. This step is important as it helps in reducing the dimensionality of the data and improving the quality of the input data. We then split the data into training and testing sets to train and evaluate our model.

We used Count Vectorizer to convert the preprocessed text data into a matrix of token counts. Count Vectorizer assigns a unique number to each distinct word or token in the text corpus and then creates a vector of counts for each document in the corpus. The resulting matrix is used as input data for the classification model. Count Vectorizer is a popular technique used in natural language processing as it allows for the transformation of text data into a numerical format that can be easily used in machine learning algorithms.

We used Linear Regression as our classification algorithm. Linear Regression is a simple yet powerful algorithm that is commonly used for regression analysis. In our experiment, we used it for classification by training the model on the training set using the Count Vectorizer output as input and the known category labels as output. The model then made predictions on the testing set. Linear

Regression has the advantage of being computationally efficient and easy to interpret, making it a popular choice in machine learning.

To find the best set of hyperparameters that optimize the model's performance we used HalvingGridSearchCV. HalvingGridSearchCV is a method for hyperparameter tuning that uses a combination of GridSearchCV and randomized search. It is designed to reduce the time and resources required to search for the best set of parameters by eliminating unpromising candidates early in the search process. Best score was achieved with $max\_df = 0.656$ and $min\_df = 0.001$ as CountVectorizer's hyperparameters and $C = 0.8$ as LinearRegression's one.

Our model achieved a precision of 0.89, a recall of 0.97, a F1-score of 0.93 and MAE of 0.275 on validation data, and 0.2624 on test data.

5.3.2 Naive Bayes

The Naive Bayes approach is a statistical learning algorithm used for text classification. The algorithm is based on Bayes' theorem and makes strong assumptions about the independence of features. The Naive Bayes approach works by calculating the probability of a document belonging to each class and then selecting the class with the highest probability. The algorithm assumes that the presence of each word in a document is independent of the presence of other words. This is a strong assumption, which is why the approach is called "naive". Despite this limitation, the Naive Bayes approach has been shown to work well in practice and is widely used in many applications.

To conduct the experiment, we first divided the dataset into a training set and a validation set, with 80% of the data used for training and 20% for testing. This division ensures that the model is trained on a large enough sample of the data to learn patterns and generalizations, while also allowing us to evaluate the model's performance on unseen data. Next, we preprocessed the dataset the same way as in the previous experiment: we removed punctuation, stop words, digits, and lemmatized and lowercased the words.

We used the Naive Bayes algorithm from the scikit-learn library to train a sentiment analysis model on the training set. We used the Bag of Words model to represent the text data as a matrix of word frequencies. This model treats each document as a bag of words, where the order of the words is not important, and only the frequency of each word matters.

We evaluated the performance of the model on the validation set using precision, recall, F1-score and MAE metrics. The results showed that the Naive Bayes algorithm achieved precision of 0.91, recall of 0.92, F1-score of 0.92, and MAE of 0.324. On the competition's test set this model achieved MAE of 0.321.

5.4 Other experiments

5.4.1 Spacy Embeddings and Gradient Boosting

Word embeddings are a powerful tool for NLP tasks. They are numerical representations of words in a high-dimensional vector space. The 'en_core_web_lg' model of the Spacy library is a pre-trained model that can be used to obtain word embeddings for text data. We used this model to obtain word embeddings for our text reviews.

Same way as in the previous experiment, we divided the dataset into 80% training data and 20% validation data and preprocessed the dataset: by removing punctuation, stop words, digits, and by lemmatizing and lowercasing the words.

After obtaining embeddings, we have chosen CatBoost Classifier as a model for gradient boosting. CatBoost is an open-source gradient boosting on decision trees library with categorical features support out of the box, successor of the MatrixNet algorithm. One of the key advantages of CatBoost is its ability to handle large datasets with high-dimensional features. This makes it particularly well-suited to NLP tasks, where the number of features can be very large. CatBoost is also capable of handling categorical features, text features and missing values, which can be a challenge for other machine learning algorithms.

As a result of the experiment, we evaluated the performance of the model on the validation set using precision, recall, F1-score and MAE metrics. The results showed that the CatBoost model trained on Spacy embeddings achieved precision of 0.84, recall of 0.88, F1-score of 0.85, and MAE of 0.309. On the competition's test set this model achieved MAE of 0.317.

5.4.2 SentenceTransformers and Gradient Boosting

In this experiment, we used sentence transformers specifically RoBERTa-Large to obtain embeddings of words. Sentence transformers are a type of deep learning model that can encode a sentence into a high-dimensional vector. These vectors capture the semantic meaning of the sentence, making them useful for a wide range of NLP tasks. RoBERTa-Large is a pre-trained sentence transformer model that has achieved state-of-the-art results on various NLP benchmarks.

Unlike previous experiments, in this experiment we did not carry out primary text processing. This means that we did not perform any preprocessing steps such as stemming, stop-word removal, or text normalization. This allowed us to evaluate the performance of RoBERTa-Large on raw text data. We then used cross-validation to evaluate the performance of the embeddings on the CatBoost Classifier model.

After calculating the metrics obtained on cross-validation, we got the following result: precision of 0.94, recall of 0.92, F1-score of 0.93, and MAE of 0.164. On the competition's test set this model achieved MAE of 0.1652. We found that the embeddings obtained from RoBERTa-Large significantly improved the performance of the CatBoost classifier and improved the MAE target metric by 52.11% compared to the previous experiment.

This suggests that RoBERTa-Large is capable of capturing the semantic meaning of words and phrases, making it a useful tool for NLP applications. Furthermore, our experiment suggests that primary text processing may not always be necessary when using advanced NLP models such as RoBERTa-Large.

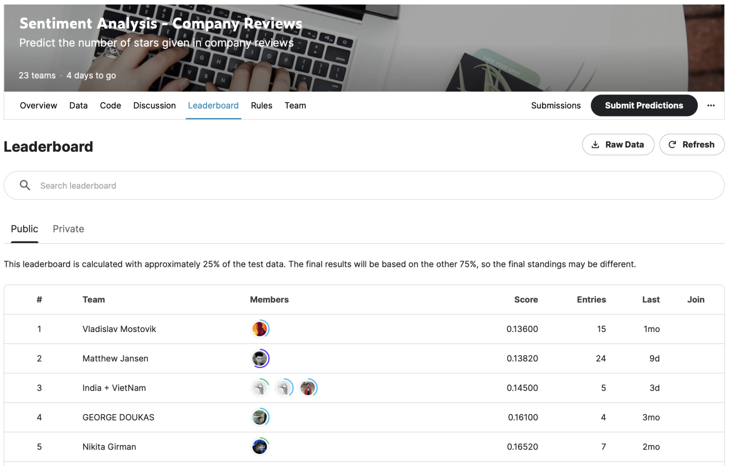
6 Results

There is a lot of approaches to try, but we have chosen to compare only four and our main approach - BERT. As we can see, BERT achieved SOTA results in this dataset, and we won the first place in this kaggle competition with MAE-value 0.136. On other metrics, such as Precision, Recall and F1-score BERT also achieved better results. In Tab. 1. you can see results of all comparable approaches.

Approach	Precision	Recall	F1	MAE
CountVectorizer + LinearRegression	0.89	0.97	0.93	0.275
NaiveBayes	0.91	0.92	0.92	0.321
SpacyEmbeddings + CatBoost	0.84	0.88	0.85	0.317
SentnceTransformers + CatBoost	0.94	0.92	0.93	0.164
BERT	0.98	0.97	0.975	0.136

Table 1: Metrics.

The best approach got the first place in public competition on Kaggle Fig 3.



This leaderboard is calculated with approximately 25% of the test data. The final results will be based on the other 75%, so the final standings may be different.

#	Team	Members	Score	Entries	Last	Join
1	Vladislav Mostovik		0.13600	15	1mo	
2	Matthew Jansen		0.13820	24	9d	
3	India + VietNam		0.14500	5	3d	
4	GEORGE DOUKAS		0.16100	4	3mo	
5	Nikita Girman		0.16520	7	2mo	

Figure 3: Leaderboard screenshot.

7 Conclusion

We carefully evaluate results from several approaches including both classic NLP methods and neural networks. We find out that such simple methods as Count Vectorizer with Linear Regression and Naive Bayes give fairly high baseline results. If text preprocessing was carried out properly, these approaches could be better than more complicated methods, for example, SpacyEmbeddings with CatBoost. However, it is still difficult for classic models to compete with complex models such as sentence transformers. Both BERT and RoBERTa demonstrated the highest results compared to other approaches. However, the best one appeared to be classic BERT model which we plan to integrate in our service.

References

[Baranyuk, 2021] Baranyuk, T. (2021). Boosted embeddings with catboost. *https://towardsdatascience.com/boosted-embeddings-with-catboost-8dc15e18fb9a.*

[Chawda, 2022] Chawda, G. (2022). Text classification using decision forests and pretrained embeddings. *https://keras.io/examples/nlp/tweet-classification-using-tfdf/.*

[Cortinhas, 2023] Cortinhas, S. (2023). Sentiment analysis - company reviews.

[Jacob Devlin, 2019] Jacob Devlin, Ming-Wei Chang, K. L. K. T. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *https://arxiv.org/pdf/1810.04805.pdf.*

[Jain, 2021] Jain, P. (2021). Basics of countvectorizer. *https://towardsdatascience.com/basics-of-countvectorizer-e26677900f9c.*

[Kamran Kowsari, 2020] Kamran Kowsari, Kiana Jafari Meimandi, M. H. S. M. L. B. D. B. (2020). Text classification algorithms: A survey. *https://arxiv.org/pdf/1904.08067.pdf.*

[Liudmila Prokhorenkova, 2019] Liudmila Prokhorenkova, Gleb Gusev, A. V. A. V. D. A. G. (2019). Catboost: unbiased boosting with categorical features. *https://arxiv.org/pdf/1706.09516.pdf.*

[Shiraly, 2023] Shiraly, K. (2023). Spacy embeddings and gradient boosting. *https://www.width.ai/post/spacy-text-classification.*

[Yinhan Liu, 2019] Yinhan Liu, Myle Ott, N. G. J. D. M. J. D. C. O. L. M. L. L. Z. V. S. (2019). Roberta: A robustly optimized bert pretraining approach. *https://arxiv.org/pdf/1907.11692.pdf.*