Project 17 Natural Language Processing and Text Mining 2023

# Restaurant Reviews Corpus 2

Trung Dang (2305577), Suhail Bashir (2307236), Mubashar Ali (2307351)

November 2, 2023

***Keywords***
Sentiment Analysis, WordCloud, BERT Embedding, Deep Learning,

## 1 Introduction

In this digital era where people are as fervent about rating a presidential candidate as they are about rating their pastry, opinions fly faster than food delivery in a traffic jam. This project is an effort to delve deep into the realm of restaurant reviews to learn common patterns in the data and utilize machine learning models to predict ratings from the reviews.

The goal of this project is two-fold. In the first part, we aim to find the common patterns in a publicly available restaurant review dataset [1]. The objective of this initial phase is to identify the underlying patterns such as distribution of words, sentiment distributions, and similarities amongst reviews. We first split the data into 5 dataframes based on the different score values of ratings (5,4,3,2,1). We analyzed the sentiments of reviews by identifying the proportion of positive and negative opinion lexicon words in each dataframe. We further explored the differences between each dataframe using empath categories [2], which helped us to identify the main categories associated with each dataframe. We analyzed the top 15 categories associated with each dataframe. All these findings helped us in understanding the difference in the patterns and distinction in the languages used in different dataframes, which further helped us to understand the subtle variation in the customer perception and expression across the 5 different ratings.

In the later part of this project, we used various machine learning algorithms to predict ratings based on the features extracted from the text of the reviews. Furthermore in the pursuit of higher performance of models, we also experimented with state-of-the-art models such as Transformers to achieve higher levels of accuracy in predicting review ratings.

## 2 Methodology

### 2.1 Data Preparation & Exploration

*A. Split Dataframes*

As mentioned for our tasks we split our dataset into 5 different dataframes based on 5 unique values of our ratings column (5, 4, 3, 2, 1). The intermediate (like 3.5) scores were ignored. After that there were a couple of null values that we removed and got our data ready for the next phase of our project which is data exploration.

*B. Common Words*

For the first data exploration task, we used WordCloud [3], which is a Python tool used for graphical representations of word frequency that gives greater prominence to words that appear more frequently in a source text. We plotted word clouds twice initially before removing stopwords and then after removing stopwords. Plots are given in the results and discussions section. In the second part of this task we restricted the contents of each dataframe to sentiment-related words only. We used a public linguistical resource [4] which has a large list of positive and negative sentiment words of the English language. For each review corresponding to each dataframe, we removed all the words except those having a positive or negative tone. After doing this we again used WordCloud to plot the contents of each dataframe. This time non-emotion words were removed and word clouds offered clearer visualization of the distinctly positive and negative aspects within the reviews.

*C. Empath Categorization*

Empath [5] is a Python based tool used for analyzing text across lexical categories. It has 200 builtin, predefined categories. We

used Empath to identify the main categories associated with each dataframe. We took the reviews of each dataframe, passed it to the *lexicon.analyze* method defined in the *Empath* class. The method returns a dictionary with keys as categories analyzed over the text and values as the normalized score of the categories. We plotted the top 15 categories corresponding to each dataframe based on the normalized score.

### *D. Sentiment Proportion*

In order to get an idea about the sentiment of dataframes we calculated the weighted average of the number of positive and negative opinion words for each dataframe. This task allowed us to emphasize the overall sentiment by evaluating the weighted density of positive and negative words, thereby providing a more balanced and refined understanding of the emotional undercurrents present in the reviews of different ratings.

### *E. Pretrained Embeddings and Relationships to Ratings*

In this experiment, we want to examine the relationship between distances of two review embeddings and their differences in user ratings. To check this relationship, firstly we need to formulate ways to compute the review embeddings. In this context, a review means the text of the user review given to a restaurant

We will compute the review embeddings based on word embeddings. We use the following word embeddings in our experiment:

- fasttext-wiki-news-subwords-300: FastText [6] word embeddings from Gensim [7], pretrained on Wikipedia dump 2017, UMBC web-based corpus, and statmt.org news dataset.
- glove-wiki-gigaword-300: Glove [8] word embeddings from Gensim, pretrained on Wikipedia dump 2014, and Gigaword 5.
- word2vec-google-news-300: Word2Vec [9] word embeddings from Gensim, pretrained on Google News.
- bert-base-uncased: BERT [10] embeddings from HuggingFace [11], pretrained on BookCorpus, and English Wikipedia dump. More information is provided in here.
- nlptown/bert-base-multilingual-uncased-sentiment: BERT model from HuggingFace, based on bert-base-uncased and finetuned on product reviews in six languages: English, Dutch, German, French, Spanish, and Italian. More information is provided in here.

The process to get review embeddings is different for non-BERT and BERT models. For FastText, Glove, and Word2Vec, we do as follows:

- We will tokenize the review using NLTK [12], configuration for English language.
- For each token, we attempt to get the corresponding embedding and collect the embedding if it exists.
- Otherwise, we skip the token that does not exist in the pretrained word embeddings.
- To get the review embedding, we compute the mean operation in the sequence dimension over all the tokens, so the review embedding will have the same size as one word embedding.

For BERT, we do as follows:

- Each BERT model in HuggingFace is packaged with a corresponding tokenizer, which we use to tokenize the review.
- We truncate the length of sequence of tokens to 512.
- We add a special CLS token at the start of the sequence.
- We run the sequence of tokens through the BERT model.
- We get the embedding of CLS token, after the pooler layer (MLP) from BERT. This is considered the output review embedding.

Provided the computation method of review embeddings, we do a small experiment to see the correspondence between embeddings and ratings. We will collect $N$ samples. For each sample, we choose a random review from the list of reviews with rating 1, rating 2, and rating 4. We compute the cosine distances between pairs of review embeddings. As both rating 1 and rating 2 are considered as negative, compared to positive sentiment of rating 4 and rating 5, we expect the distance between reviews of rating 1 and rating 2 will be smaller than reviews of rating 1 and rating 4. Therefore, when this expectation is fulfilled, the sample is considered as a hit, which means the embeddings really show the user rating. After collecting all samples, we compute the hit rate as:

$$HitRate = HitCount/N \tag{1}$$

We repeat such trial $K$ times. In this experiment, $N = 200$ and $K = 5$.

*F. Embeddings of Ratings*

In this experiment, we want to compare the representative embeddings of the user ratings (from rating 1 to rating 5). The process to compute the representative embeddings is as follows:

- We use two word embeddings "bert-base-uncased" and "nlptown/bert-base-multilingual-uncased-sentiment" from the previous section.
- We compute the review embeddings similar to the process in the previous section.
- For each rating, we collect all the reviews of such rating score. Then, the rating embedding is computed by executing the mean operation over all review embeddings of one rating.

Then, we compute the cosine distances between each pair of rating embeddings. The cosine similarity is computed using:

$$CosineSimilarity(x, y) = max(1 - CosineDistance(x, y), 0) \quad (2)$$

## 2.2 Predictive Modelling

*A. Machine Learning Models*

To build ML models that can learn to predict user ratings based on text of review, we formulated the problem as a classification problem. The 5 unique values of the rating column were taken as 5 classes for our problem. We used tf-idf vectorizer to process the text data into suitable representation for our ML models. We tested the different features of n-grams from unigram all the way to the 5-gram features.

For each of the values of n-grams, we trained two machine learning models: random forest classifier [13], and gradient boosting classifier [14]. The implementation was carried out using the scikit-learn library [15] in Python. Models were trained using 80% of data for training and 20% for testing. Two metrics, accuracy ( which is ratio of correctly predicted instances to the total number of instances) and F1-score (which is the harmonic mean of the precision and recall of a classification model), were used to evaluate the models. The formula used for calculating accuracy and F1-score are given as follows:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (3)$$

$$F1 - Score = \frac{(2 \times precision \times recall)}{(precision + recall)} \quad (4)$$

$$TP : TruePositive \quad (5)$$

$$FP : FalsePositive \quad (6)$$

$$TN : TrueNegative \quad (7)$$

$$FN : FalseNegative \quad (8)$$

$$precision = \frac{TP}{(TP + FP)} \quad (9)$$

$$recall = \frac{TP}{(TP + FN)} \quad (10)$$

*B. Deep Learning Models*

In this section, we want to experiment with different neural network architectures for the task of rating prediction. The input is the text of a review, and the model predicts a rating score from 1 to 5. All of our methods follow the same paradigm as follows:

- The review text is tokenized by a tokenizer that is provided along with the BERT or RoBERTa model. We truncate at max length 512.
- The token embeddings are computed. CLS token is used.
- An additional neural network is constructed on top of the output token embeddings. It may use one or all the token embeddings.
- After that, a set of linear layers will output the logits corresponding to difference ratings, as well as other supplementary outputs. We formulate this problem as a classification problem.

We use two models for token embeddings:

- nlptown/bert-base-multilingual-uncased-sentiment: BERT model from HuggingFace, based on bert-base-uncased and finetuned on product reviews in six languages: English, Dutch, German, French, Spanish, and Italian. More information is provided in here.
- cardiffnlp/twitter-roberta-base-sentiment-latest: RoBERTa [16] model from HuggingFace, trained on Twitter sentiment data. More information is provided in in this link. .

We use the following neural models on top of the token embeddings:

- No downstream model at all. The classification linear layers are connected to the pooler output of BERT or RoBERTa model.
- MLP layers after the pooler output of BERT or RoBERTa model. This pooler output only consists of the embedding of CLS token.
- LSTM [17] and TextCNN [18] over embeddings of the entire sequence of tokens. We use two LSTM and two TextCNN layers in an interleaving manner.
- A Transformer encoder [19] with two layers, attending to embeddings of the entire sequence of tokens.

We use binary cross entropy loss over 5 classes, corresponding to 5 ratings. We also use two supplementary losses to guide the model in rating prediction and add structure to the rating:

- sentiment classification: neutral (3), negative (1 and 2), positive (4 and 5).
- strength classification: strong (1 and 5), weak (others).

The dataset is very imbalanced in terms of class distribution. We use two methods to tackle this issue:

- In the rating classification loss, we add weighting by class distribution, so rare classes will have a stronger impact on the loss function.
- We collect all the reviews of each rating and re-sample the dataset. The number of samples $S$ is the same for every class. For each sample of one class, we randomly choose a review from that class as the training instance. One iteration over the dataset will go through 5 classes $S$ times, so the number of steps in one epoch will be $5S$. We set $S = 1600$.

### *C. Large Language Models*

In this section, we want to experiment with large language model and prompting for the task of rating prediction. The input is the text of a review, and the model predicts a rating score from 1 to 5.

The large language model that we use is Vicuna 7B [20], which is provided in https://github.com/lm-sys/FastChat.

We use few shot prompting as a guideline for the model to understand the task. Then, based on the task description, the model needs to generate its answer.

We add some reasoning to the few-shot examples, as a way to simulate chain-of-thought prompting. Then, the model is asked to generate its reasoning and explanation for its final rating output.

Based on the generated answer from the large language model, we parse the answer and get only the first number as the final rating prediction. If this number is from 1 to 5, we accept the answer. Otherwise, we consider this as a nonsense output and return a default prediction, which is 3 the neutral rating.

The full prompt of our method is provided here, which consists of 5 examples. Each example is a review text, a corresponding user rating, and some handmade explanation.

---

```
Based on the review, please
    provide a prediction on the
    rating given by the reviewer to
     the restaurant. Some examples
    of rating predictions and
    reasons are provided. Let's
    think step by step.

#
Review: Worst place to have a
    sizzler. Tried hyderabadi
    sizzler which was just rotten
    chicken biriyani on a hot plate
    . The noodles in the veg
    sizzler were very bad as well.
    These guys don't know how to
    make a sizzler. The staff was
    also not well trained.

Rating: 1

Reason: The reviewer dislikes both
     the food and the staff. They
    use very strong language to
    condemn the place, without any
    praise. There is no good so it'
    s a 1.

#
Review: The ambience is good. But
    the food wasn't up to the mark.
     The non veg starters were
    horrible. Full of bones. The
    veg starters were okay. The
    variety was less. The main
    course was okay. There was
    variety but no taste. There
    were hardly any desserts.
    Disappointed.
```

```
Rating: 2

Reason: The ambience is praised.
    However, the food is criticized
     very badly and the reviewer is
     disappointed. So it's a 2.

#
Review: The quantity was great but
     the mutton pieces were just 4
    numbers out of which 3 were
    bones as well. Only average
    impression to offer. Hope the
    will keep the standard better.

Rating: 3

Reason: There are good and bad
    points in the review. Overall
    the place is considered average
    , which is a 3.

#
Review: Good one. You get a
    combination of south and north
    Indian varieties in a single
    thali. The servicing here is
    very slow. I have been here
    twice and it took more than
    half an hour to get the
    starters. Staff are polite and
    friendly.

Rating: 4

Reason: The impression from the
    review is good in general. They
     like the food and the staff.
    Only the speed of servicing is
    awful. Therefore it's a 4.

#
Review: The food was excellent. I
    had a fast today and this kind
    was food was awesome . I had a
    stomach full meal and the
    experience was really well .
    Thank you fasoos . The paneer
    was really well and the rotis
    were very very good .

Rating: 5

Reason: They like everything about
     the place, so the review of
    full of praises. No aspect is
    condemned. They also thank the
    restaurant. This is a 5.

#
```

# 3 Results & Discussions

*A. Common Words*

Three word clouds were plotted: before stopword removal, after stopword removal, and finally after removing non-emotional words.

A word cloud without the removal of stopwords often fails to provide meaningful insights due to the inclusion of common and frequently occurring words that lack distinct significance. The resulting visual representation becomes cluttered and fails to highlight the essential or sentiment-bearing terms within the text as shown in the figure 1.

After the removal of stopwords, the word cloud presented a clearer and more focused visualization, emphasizing the most relevant and sentiment-rich terms present in the text as shown in the figure 2. However, there are words belonging to the common topic (related to food and restaurants) of the reviews in every word cloud.

After keeping only sentiment words non-sentiment words which were very dominant earlier such as *'place'*, *'ambience'* were now removed. This filtering process resulted in a focused representation, showcasing terms that distinctly conveyed either positive or negative sentiments, offering a more targeted visualization of the emotional tonality inherent in the text. This plot is available in figure 3.

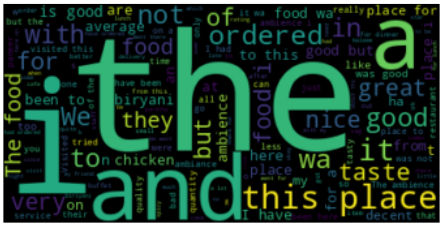
The initial insights that can be inferred from WordCloud representations for five different dataframes are that as we go to lower ratings, the negative words like *'worst'*, *'bad'* start becoming more prominent. There were also positive words like *'good'* present in the lower ratings. The reason is that the negation of those words such as *'not'* for *'not good'* was removed by the stopword removal. Other insights from reading the reviews from lower ratings was that even for low ratings such as rating '1' there used to be one positive sentence such as *'ambience was **good** but food was really **bad**'*.

*B. Empath Categories*

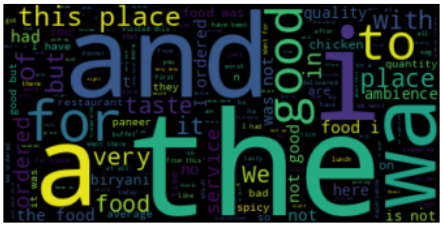
The top-15 empath categories for each dataframe based on the normalized score were plotted as shown in the figure 4. The dataframes of rating '5' and '4' have the same top five categories which are *'friends'*, *'positive_emotion'* 'politeness', 'achievement',

(a) Rating 5.

(b) Rating 4.



(c) Rating 3.



(d) Rating 2.

(e) Rating 1.

Figure 1: WordCloud without stopword removal.

(a) Rating 5.

(b) Rating 4.

(c) Rating 3.

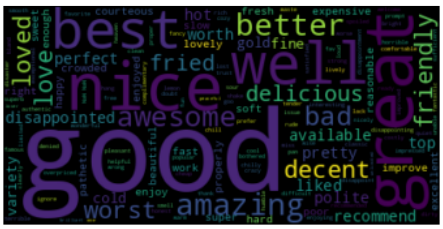
(d) Rating 2.

(e) Rating 1.

Figure 2: WordCloud After stopword Removal.


(a) Rating 5.


(b) Rating 4.




(c) Rating 3.


(d) Rating 2.


(e) Rating 1.

Figure 3: WordCloud having only sentiment words.

'optimism'. For the dataframe of rating '3' top five categories are *'positive_ emotion', 'friends', 'politeness', 'negative_ emtion'*. For the dataframe of rating '2', the top five categories are *'negative_ emotion', 'positive_ emotion', 'friends', 'politeness', 'optimism'*. For the dataframe of rating '1' the top five categories are *'negative_ emotion', 'positive_ emotion', 'friend', 'hate', 'emotional'*.

This analysis showcased a compelling correlation between lower ratings and the heightened prevalence of empath categories associated with negative emotions. Insights from empath categorization is very intuitive, which is that when we go to lower ratings the negative word categories start becoming more prominent.

### *C. Sentiment proportion*

The total count of positive and negative words in five separate dataframes was calculated. To acquire a normalized perspective, we divided the tally of both positive and negative terms by the total number of words present within each respective dataframe. The resulting normalized values represented the density of positive and negative expressions within the text data.

Upon plotting these outcomes as shown in the figure 5, an intuitive and compelling pattern emerged: as we traversed towards dataframes associated with lower ratings, the prominence of negative words increased notably. This observation highlighted a proportional rise in the presence of negative sentiments as the ratings decreased, indicating a distinct shift towards a more negative tone within the reviews.

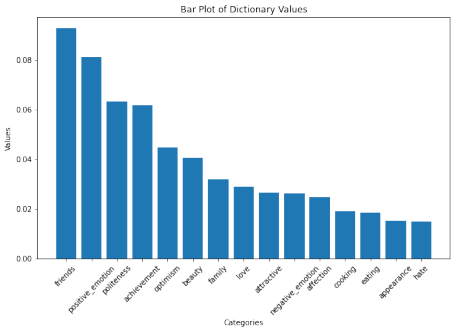
### *D. Pretrained Embeddings and Relationships to Ratings*

We will show the results of our experiment. However, this experiment has low reproducibility, so repeating the same steps may get different results:

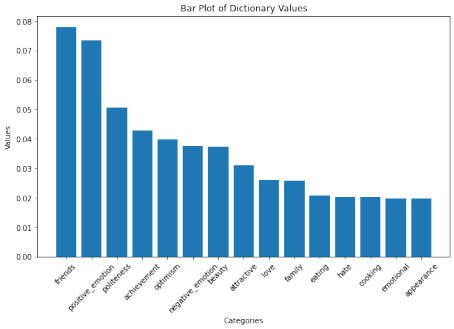
| **Embedding** | **Hit Rate (%)** |
|---|---|
| fasttext-wiki-news-subwords-300 | 49.2 ± 4.389 |
| glove-wiki-gigaword-300 | 49.4 ± 3.655 |
| word2vec-google-news-300 | 53.2 ± 5.272 |
| bert-base-uncased | 52.5 ± 3.669 |
| nlptown/bert-base-multilingual-uncased-sentiment | 92.6 ± 0.678 |

Table 1: Results of hit rate

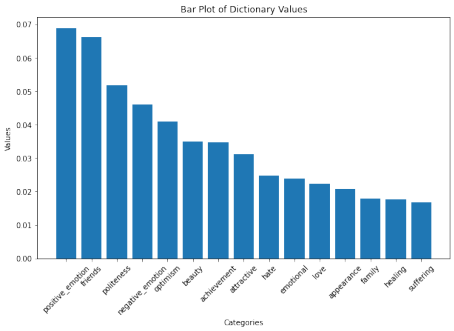
We can see that, without finetuning on sentiment dataset, cosine distances between embed-



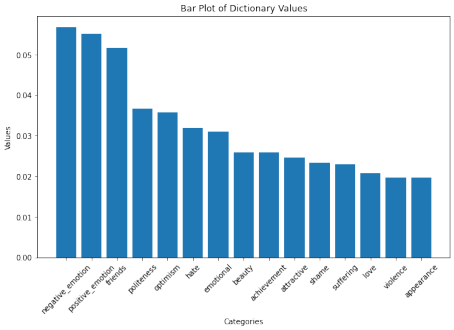

(a) Rating 5.



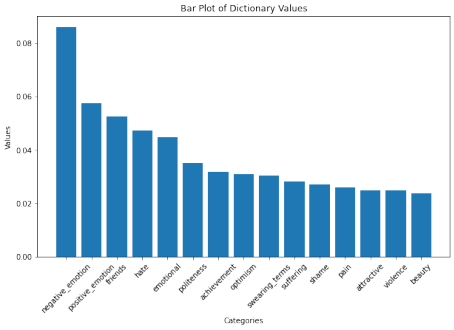

(b) Rating 4.




(c) Rating 3.

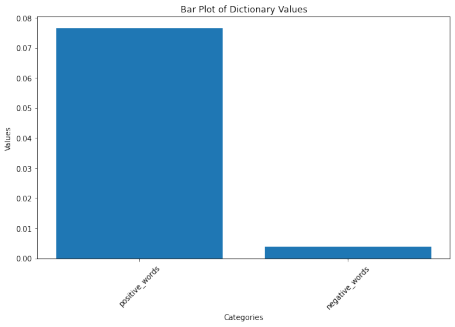



(d) Rating 2.

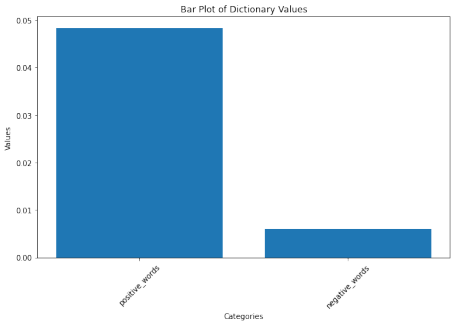



(e) Rating 1.

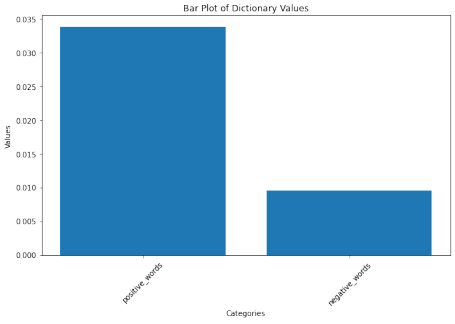
Figure 4: Empath Categories.



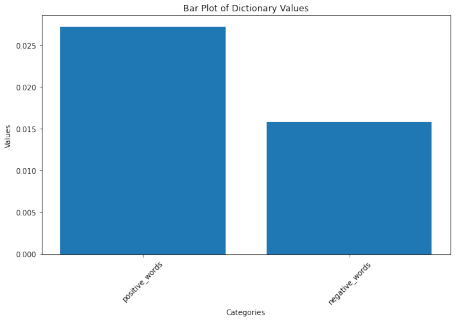

(a) Rating 5.



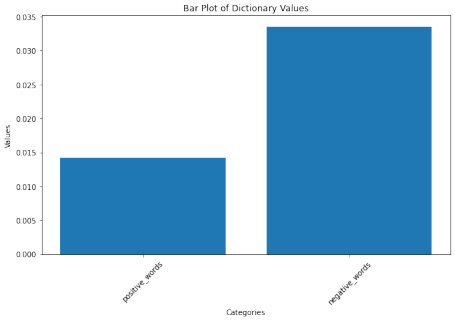

(b) Rating 4.




(c) Rating 3.




(d) Rating 2.




(e) Rating 1.

Figure 5: Positive and negative words proportion.

dings do not translate well into proximity in ratings. This is because there are many "dimensions" to the text apart from its positive or negative sentiment. For example, two reviews discussing food will have lower cosine distance than one about food and one about service, so the "topic" is also possibly one dimension of the embeddings in addition to the "sentiment". Therefore, when pretrained on generic data, the embeddings do not show well the difference in sentiment only, because they need to capture other aspects of the text.

However, when finetuned on data of product reviews, the correspondence between cosine distance and rating difference in BERT is stronger. As a result, the hit rate is significantly higher than other models.

*E. Embeddings of Ratings*

The results for bert-base-uncased are:

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 1.0 | 0.9970 | 0.9907 | 0.9819 | 0.9758 |
| 2 | 0.9970 | 1.0 | 0.9971 | 0.9892 | 0.9781 |
| 3 | 0.9907 | 0.9971 | 1.0 | 0.9963 | 0.9829 |
| 4 | 0.9819 | 0.9892 | 0.9963 | 1.0 | 0.9874 |
| 5 | 0.9758 | 0.9781 | 0.9829 | 0.9874 | 1.0 |

Table 2: Matrix of similarity between ratings

The results for nlptown/bert-base-multilingual-uncased-sentiment are:

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 1.0 | 0.8664 | 0.3414 | 0.0000 | 0.0000 |
| 2 | 0.8664 | 1.0 | 0.7142 | 0.0000 | 0.0000 |
| 3 | 0.3414 | 0.7142 | 1.0 | 0.6232 | 0.2605 |
| 4 | 0.0000 | 0.0000 | 0.6232 | 1.0 | 0.8914 |
| 5 | 0.0000 | 0.0000 | 0.2605 | 0.8914 | 1.0 |

Table 3: Matrix of similarity between ratings

In general, the cosine similarity between close ratings is higher than that of far ratings. This applies for both BERT models. However, one exception happens with rating 4 using “bert-base-uncased”, where its embedding is roughly equally close to that of rating 2 and rating 5.

Although there are differences between cosine similarity scores when using “bert-base-uncased”, these differences are relatively small compared to using “nlptown/bert-base-multilingual-uncased-sentiment”. We suspect that most reviews can be considered to be of the same topic (i,e restaurant reviews), so they are typically close in that dimension. On the other hand, the differences are clear when using embeddings finetuned on review data, where magnitude can be as large as 0.89.

*F. Machine Learning Models*

We employed two distinct machine learning models Random forest classifier and Gradient boosting classifier. The models were trained and tested on a diverse range of n-gram values, varying from unigrams to 5-grams to determine their influence on the predictive accuracy. Notably, the analysis revealed that the 4-gram value exhibited the most promising performance in both classifiers as can be seen in the table 4 and 5.

| | N-gram Value | Accuracy | F1 Score |
|---|---|---|---|
| 1 | 1 | 48.72% | 42.17% |
| 2 | 2 | 56.47% | 50.01% |
| 3 | 3 | 60.44% | 54.29% |
| 4 | 4 | 61.52% | 55.49% |
| 5 | 5 | 60.85% | 54.58% |

Table 4: Results for Random Forest Classifier

| | N-gram Value | Accuracy | F1 Score |
|---|---|---|---|
| 1 | 1 | 49.54% | 46.60% |
| 2 | 2 | 57.44% | 55.69% |
| 3 | 3 | 61.72% | 60.01% |
| 4 | 4 | 62.69% | 60.72% |
| 5 | 5 | 61.72% | 59.55% |

Table 5: Results for Gradient Boosting Classifier

As the n-gram value increased, each model’s accuracy and robustness in rating predictions underwent a discernible shift. The experimentation with different n-gram sizes shed light on the significance of capturing nuanced linguistic patterns in the text data. Notably, the 4-gram configuration stood out as the most optimal, demonstrating a balance between capturing contextual information and avoiding excessive data sparsity.

While both models showed promise, Gradient Boosting was marginally more successful than Random Forest in its predictive capabilities. Throughout our analysis, the models excelled in discerning the polarity of the reviews i,e identifying whether the sentiment is positive or negative. However they encountered challenges in more granular classification as it is evident from the confusion matrices in figure 6 and 7 that the majority of mis-classifications occur between the ratings 4 and 5. This indicates there is a notable challenge for these models in distinguishing between these closely adjacent rating

categories.

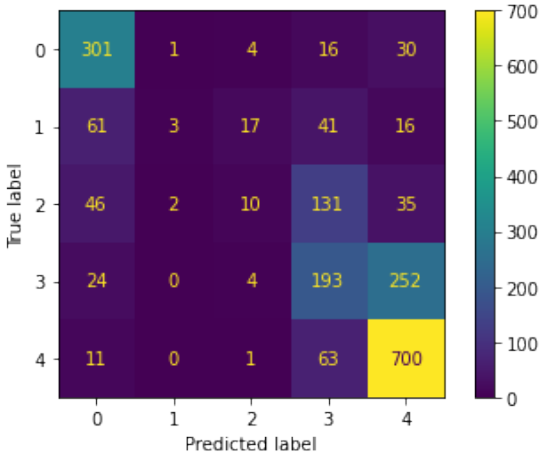



Figure 6: Random Forest confusion matrix

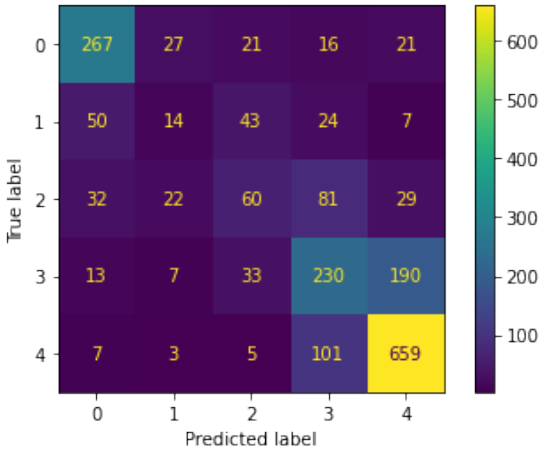



Figure 7: Gradient Boosting confusion matrix

### *G. Deep Learning Models*

The evaluation results of different neural models are as follows:

| Model | Accuracy | F1-score |
|---|---|---|
| BERT+classifier | 70.43% | 70.56% |
| RoBERTa+classifier | 69.97% | 69.64% |
| BERT+MLP | 79.54% | 79.50% |
| RoBERTa+MLP | 75.62% | 75.77% |
| BERT+LSTM+TextCNN | 92.57% | 92.55% |
| RoBERTa+LSTM+TextCNN | 92.57% | 92.51% |
| BERT+Transformer | 92.62% | 92.61% |
| RoBERTa+Transformer | 93.18% | 93.14% |

Table 6: Evaluation of deep learning methods

Based on these experiment results, we can see that more complex models are better for this classification task. When we use MLP on top of the base model, the performance is improved, which means the MLP can model more features from the base embeddings, without overfitting the training data. As a results, the model can predict more accurately with higher-level features from the MLP.

When we move from using one CLS token only to using all the tokens in the sequence, the performance is improved significantly. The features from CLS token only may not be enough to model the sentiment of the entire review, so using more information from other tokens helps the model to classify better. It also shows the effectiveness of our two sequence modeling modules.

Between LSTM+TextCNN and Transformer modules, the Transformer module is slightly better in terms of accuracy and F1-score.

Between two pretrained embedding models, one of them is not strictly better than the other. The difference in their performance depends on the downstream architecture that we employ.

Overall, the setting with the highest accuracy and F1-score in our experiments is the RoBERTa+Transformer model. We want to show the confusion matrix of this model:

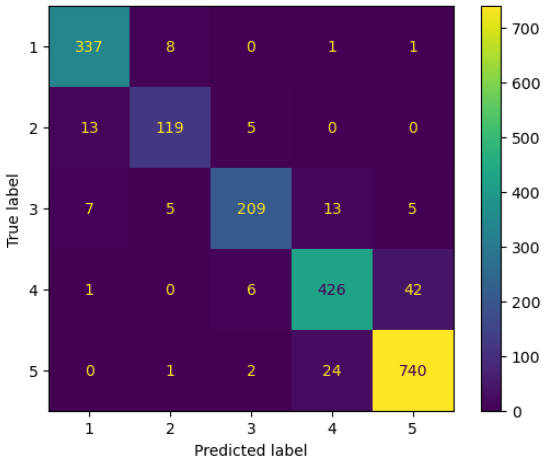



Figure 8: RoBERTa+Transformer confusion matrix

Since the performance of the model is good, we can see the diagonal of the matrix has highest values compared to other elements in a row. Looking into the failure cases, we see that most of the mistakes are in telling apart the rating 4 and rating 5. This is expected, because close ratings are more difficult to classify than far ratings.

### *H. Large Language Models*

The evaluation results of this method are as follows:

- Accuracy: 60.81%
- F1-score: 60.19%

Although the Vicuna model has been trained on a massive amount of text data, it is still difficult to reach the performance of dedicated fine-tuned models. Another problem with this approach is that the output is free-form, so the model is not guaranteed to return a well-defined rating score.

## Conclusions

In conclusion, our exploration of restaurant review data revealed insightful patterns, notably the prominence of negative words correlating with lower ratings. The prevalence of negative sentiment increased notably as ratings decreased. Moreover, our in-depth analysis of machine learning models, particularly the Random Forest Classifier and the Gradient Boosting Classifier, shed light on the influence of n-gram values in text analysis. Notably, the use of different n-gram sizes, ranging from unigrams to 5-grams, demonstrated varying impacts on model performance. The 4-gram configuration emerged as the most successful, balancing contextual relevance with manageable complexity.

Despite the overall success of these models in distinguishing positive from negative sentiments, the models faced challenges in fine-grained rating predictions. Specifically, differentiating between adjacent ratings such as 4 and 5 proved notably more intricate. To tackle this challenge, we do various experiments with state-of-the-art methods such as pre-trained large language models and embeddings, recurrent neural networks, convolutional neural networks, self-attention, few-shot learning with prompting, etc. Overall, our best-performing model can reach over 93% in accuracy and F1-score for the task of fine-grained rating prediction.

For more information regarding our project, please visit our code repository as well as the public data folder. The link to our GitHub repository is https://github.com/Suhail-BW/nlp_1st_sem. The model weights and some other data can be found in here.

## References


[1] Joakim Arvidsson. Kaggle restaurant reviews.

[2] Ethan Fast, Binbin Chen, and Michael S Bernstein. Empath: Understanding topic signals in large-scale text. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, pages 4647–4657, 2016.

[3] Andreas Mueller. word_cloud.

[4] Bing Liu and Minqing Hu. Opinion mining, sentiment analysis, and opinion spam detection. *Dosegljivo: https://www. cs. uic. edu/ liub/FBS/sentiment-analysis. html# lexicon.[Dostopano 15. 2. 2016]*, 2004.

[5] Ethan Fast. empath-client.

[6] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.

[7] Radim Řehřek, Petr Sojka, et al. Gensim—statistical semantics in python. *Retrieved from genism. org*, 2011.

[8] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[11] Shashank Mohan Jain. Hugging face. In *Introduction to Transformers for NLP: With the Hugging Face Library and Models to Solve Problems*, pages 51–67. Springer, 2022.

[12] Steven Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72, 2006.

[13] Steven J Rigatti. Random forest. *Journal of Insurance Medicine*, 47(1):31–39, 2017.

[14] Monika, Munish Kumar, and Manish Kumar. Xgboost: 2d-object recognition using shape descriptors and extreme gradient

boosting classifier. In *Computational Methods and Data Engineering: Proceedings of ICMDE 2020, Volume 1*, pages 207–222. Springer, 2021.

[15] Oliver Kramer and Oliver Kramer. Scikit-learn. *Machine learning for evolution strategies*, pages 45–53, 2016.

[16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[18] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[20] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.