Predicting Restaurant Ratings from Customer Reviews

Yunseo Choi (2023195100)

1 Introduction

In this mini project, I built a model that uses a small AI pipeline to predict how many stars a customer would give to a restaurant, based on their written reviews. Nowadays, there are many platforms such as Google Maps and Yelp that customers can leave reviews on, and estimating ratings from those text data can be useful in finding the mapping mechanism between text data and one's degree of satisfaction.

Rather than training a large model from scratch, I carefully followed the workflow of designing an AI pipeline: defining the task, choosing a naive baseline and a AI pipeline, and making comparisons between those two models. The project is intentionally small—I collected a tiny dataset of 50 restaurant reviews near University of California San Diego and built both a keyword-based baseline and a Hugging Face–based pipeline to predict 1–5 star ratings. The main goal is to get a sense of the flow of designing an AI pipeline model, rather than achieving a state-of-the-art performance.

2 Task Definition

- **Task description:** The model's task is to receive a restaurant review as an input and estimate the rating (from 1 to 5 stars) that the customer would give to the restaurant.
- **Motivation:** Learning the mapping between the review text and numeric ratings is interesting because the mapping mechanism identifies the words that are influential in estimating customers' satisfactions. For a practice use, such a system could easily summarize large numbers of reviews, detect outlier opinions, or monitor changes in user satisfaction over time.
- **Input / Output:** The input is a single restaurant review written in English, in a free format. The desired output is an integer rating in $\{1, 2, 3, 4, 5\}$ representing the predicted number of stars that the customer assigned.
- **Success criteria:** Ideally, the system should predict the exact rating for a large fraction of test examples. To set a clear criteria, I consider the model to be successful if it makes correct predictions of the true rating for more than 50% of the time. Since this is a small project with only 50 examples, I also considered the mean squared error (MSE) to measure how far the predictions are from the real ratings.

3 Methods

This section describes both the naive baseline and the improved AI pipeline. Both methods share the same dataset and preprocessing, but they differ in how they map the review text to a rating.

3.1 Naïve Baseline

Your Baseline

- **Method description:** The baseline is a simple keyword-based model. I first defined a small list of positive keywords (e.g., ``amazing'', ``great'', ``delicious'', ``excellent'', ``friendly'', ``good'') and a small list of negative keywords (e.g., ``worst'', ``rude'', ``terrible'', ``awful'', ``disgusting'', ``overpriced'', ``bland'', ``cold'', ``fake''). For a given review, I lower-cased the text and counted how many distinct positive and negative keywords appeared at the review. Counting for duplicate words were disregarded for the sake of the model's simplicity. If positive words clearly dominate (e.g., appears at least twice more than negatives), the model predicts 5 stars; if positives are slightly more, it predicts 4 stars; if negatives clearly dominate, it predicts 1 star; if negatives are slightly more, it predicts 2 stars; and if the counts are equal or zero, it predicts 3 stars. In other words, the rating is determined by the relative balance between positive and negative keywords.
- **Why naïve:** This baseline ignores word order, sentence structure, and context. It treats all positive (or negative) keywords as equally strong signals, and it does not distinguish between different intensities of sentiment. It also assumes that the presence of a small hand-crafted set of words is enough to understand the review, which is clearly different from the human's thinking process. For these reasons, the model is naive with limitations in practical settings.
- **Likely failure modes:** The baseline can fail when reviewers use words that are not in the keyword lists, even if their review stance is evident. It also fails when the keywords are used in non-evaluative ways. For instance, if a customer writes "the cold brew I had was good", the word ``cold'' will be counted as a negative keyword even though it is just a part of a drink's name. Similarly, the model is not adept for detecting sarcasms, as it wouldn't look at the context that the word is being used. Because repeated occurrences of the same keyword only count once, the model also cannot distinguish between moderately positive and extremely positive reviews that use the same word multiple times.

3.2 AI Pipeline

Your Pipeline

- **Models used:** For the AI pipeline, I used a pre-trained transformer-based text model from the Hugging Face `transformers` library via the `pipeline` API. Specifically, I used the `Festooned/Multilingual-Restaurant-Reviews-Sentiment` model from Hugging Face Hub [1]. The model is a rating predictor that maps a review to a continuous score ranging from one to five. After the mapping, the number was rounded to a integer value between one and five.
- **Pipeline stages:**
  1. **Preprocessing:** Each raw review string from the Excel file is cleaned by removing unnecessary newlines and tabs, converting to lowercase, and collapsing multiple spaces into one. I also dropped any empty rows and ensured that the ratings are integers between 1 and 5.
  2. **Representation / embedding:** The cleaned review is passed to the Hugging Face text-classification pipeline. Internally, the pipeline tokenizes the text and feeds it into a transformer encoder, which produces a rich semantic representation (embedding) of the review. I did not access the embedding directly, but conceptually this is the representation stage.
  3. **Decision:** The model outputs a label and a score, and the label can be disregarded in our case. With the continous scalar score, I rounded it to the nearest integer and clipped

it to the range 1–5. This mapping from model output to a discrete star rating is my decision stage.

- **Design choices and justification:** I chose a pre-trained transformer model because my dataset is very small (only 50 examples), which makes it hard to train a model from scratch. Using the Hugging Face pipeline allows me to focus on the pipeline structure and evaluation rather than optimization details. The rounding-and-clipping decision rule is simple but makes the output directly comparable to the conventional 1–5 ratings. Finally, using the same preprocessing method for both the baseline and the AI pipeline ensures a fair comparison.

4 Experiments

4.1 Datasets

Your Dataset Description

- **Source:** I made my own dataset of restaurant reviews and ratings. I manually collected 50 review–rating pairs from Google Maps for restaurants near UC San Diego. For each star rating from 1 to 5, I selected 10 reviews, so the dataset would go through a balanced evaluation.
- **Total examples:** The final dataset contains 50 examples, each consisting of a text review and the corresponding 1–5 star rating.
- **Train/Test split:** I split the dataset into training and test sets using `train_test_split` from `scikit-learn`, with a test size of 0.3 (i.e., 30% test, 70% train) and random seed 42. This divided the dataset into 35 training datasets and 15 test datasets. Because the dataset is very small, I did not create a separate validation set and used the test set for the final evaluation.
- **Preprocessing steps:** Each review was converted to a lowercase string, and tabs and newlines were replaced with spaces. The rating column is cast to an integer type, and these preprocessing mechanisms were applied before passing the data to both the baseline model and the Hugging Face pipeline.

4.2 Metrics

I evaluated both models using two metrics: accuracy and mean squared error (MSE). Accuracy measures the fraction of test examples that produces a prediction that matches the ground-truth rating. This is one of the common metrics for classification, but it does not show how close the predictions are to the real values.

Because ratings are numeric values that have orders, I also used mean squared error for comparison:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2,$$

where y_i is the ground-truth rating and $\hat{y}_i$ is the predicted rating for example i. MSE captures how far, on average, the predictions are from the true ratings, penalizing the predictions that are off by multiple stars. I found MSE more useful than the accuracy, since in a practical setting, the bigger prediction gap (e.g., predicting the customer's rating as 5, extremely satisfactory, while the real rating is 1, extremely unsatisfactory), should be penalized more than the smaller ones (e.g., predicting the customer's rating as 5, extremely satisfactory, while the real rating is 4, satisfactory).

4.3 Results

Method	MSE	Accuracy
Baseline (keywords)	2.467	0.267
AI Pipeline (HF model)	1.333	0.267

On the 15-example test set, the naive keyword baseline achieves an accuracy of 0.267 and a mean squared error of 2.467. The Hugging Face–based AI pipeline has the same accuracy of 0.267, while it has a lower MSE of 1.333, meaning its predictions are closer to the true ratings on average. So even though their performance looks similar in terms of accuracy, the AI pipeline is better at making closer predictions.

Qualitative examples. To better understand the behavior of the two systems, I look at a few individual test examples.

1. **Example 1 (mixed but disappointed):**

   > "i've been a regular customer since before this restaurant changed ownership. i'm a big fan of having a local restaurant in our university city community. during my last visit to the restaurant i purchased a shawarma to-go. i was sadly surprised when i got home to see that it had very little meat. when i got home i called the restaurant to let them know. it's disappointing to see food quality change and i hope this local restaurant can address this issue. during this time of rising prices i expect to pay more but i also expect the same quality."

   The true rating is 2 stars. The baseline predicts 3 stars (moderate), while the AI pipeline predicts 2 stars exactly. The baseline likely didn't capture any sentiment keyword, while the pipeline captures the main complaint about declining quality of the food and low meat content.

2. **Example 2 (clearly negative experience):**

   > "no i wouldn't recommend it the food was not good and the hostess did not care for my satisfaction. i told her that my chicken was cheery and she responded with "i don't know i'm not the cook". i know your not the cook but i was not the only one who returned the food, the proper response could of been "sorry for that, i'll let the cook know" ... that's probably why this place it's soo empty. sorry guys i wouldn't come again."

   The true rating is 1 star. The baseline predicts 4 stars, which is extremely satisfactory. This happens because the keyword list does not fully capture the long, detailed complaint and only focuse on keywords like "good." On the contrary, the AI pipeline predicts 1 star, correctly recognizing the content and the negative sentiment of the review.

3. **Example 3 (clearly positive but not over-the-top):**

   > "chill local vibes in this patio brewpub. the lamb chops were the standout of our meal followed by their seasoned fries. although we didn't try them, their burgers were a popular choice at tables all around us."

   The true rating is 4 stars. Both the baseline and the AI pipeline predict 3 stars. Likewise to the first example, the baseline didn't capture any keyword, because its keyword database is not that massive and inclusive. For the AI pipeline model, it produces a close estimation of the real rating by reading the context, but underestimates the strength of the satisfaction. This suggests that both methods can make slightly wrong predictions when the customer contains a small number of sentiment keywords, that are not obviously positive or negative.

Overall, these examples illustrate why the AI pipeline can have a lower MSE despite having lower accuracy: although the accuracy is low as the baseline, the predictions made by the AI pipeline model tend to be less far off from the ground-truth ratings.

5 Reflection and Limitations

Your Reflection

This mini project showed me that even a very simple AI pipeline can be built easily by borrowing code from an external source, while making a high-performance model is a different topic for discussion. I initially expected the Hugging Face model to maintain an accuracy higher than 0.5, and outperform the baseline accuracy. However, the accuracy was nearly the quarter of the whole test dataset, and the same accuracy to its baseline model. Introducing MSE metrics provided a new perspective on "closeness," and showed that the AI pipeline model can make better guesses than the baseline model. This realization showed me the importance of the metrics and made me carefully consider which metrics would be useful in illustrating the performance of different AI modeling tasks.

One possible reason for the AI pipeline's low accuracy is due to the fact that the original Hugging Face model was trained on data that contained the title and the review text. Since titles are generally short, reviewers tend to give their one-sentence sentiment on it, like "Never Going Back," or "The Hidden Gem of the Town." On the other hand, the dataset used in this experiment didn't contain any titles, and the lack of title information seems to cause in low performance of the model. Therefore, future improvements can be made by fine-tuning the model to grasp review information better, and rely on the title information less.

References

[1] Festooned. Multilingual restaurant reviews sentiment. https://huggingface.co/Festooned/Multilingual-Restaurant-Reviews-Sentiment, 2024. Accessed: 2025-12-14.