1 Introduction

1.1 Dataset

We selected a dataset of reviews that was scraped from [Google Local for various restaurants](). The dataset includes 87,013 reviews from 29,596 users for 27,896 unique businesses. Each entry in this dataset contains the following fields:

- **business_id**: A unique identifier for the business.
- **user_id**: A unique identifier for the user who wrote the review.
- **rating**: The rating given by the user (on a scale of 1 to 5).
- **review_text**: The text of the review.
- **pics**: A list of pictures associated with the review.
- **history_reviews**: A list of previous reviews written by the same user.

1.2 Exploratory Data Analysis

In our data analysis, we aim to uncover patterns and relationships within the dataset such as the distribution of ratings and the influence of review content. We will incorporate data visualization techniques such as bar charts, histograms, and word clouds to further our understanding behind possible correlations within our dataset.

```
Rating
Statistics:
count
87013.000000
mean
4.465252
std
0.833755
```

To grasp a basic understanding of the dataset, we began by looking at the mean and standard deviation of ratings in our dataset, which is from 4.465 and 0.834 respectively. This indicates that, on average, users tend to give positive reviews to restaurants as ratings closer to 5 represent higher satisfaction. The low standard deviation informs us that the ratings are near the mean, showing that user ratings for restaurants are consistently positive, with some possible extremes. We can further see this in Figure 1, a bar graph showcasing the frequency of each rating. We have a right skewed normal distribution, where the number for 5 star ratings are significantly higher than 1, 2, and 3 star ratings.

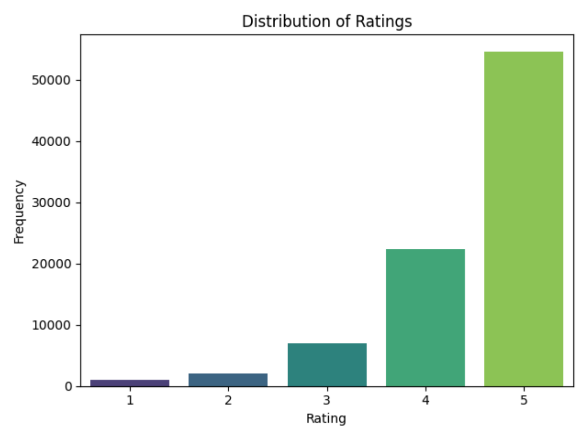

Figure 1

To gain a better understanding of the content of the reviews and its sentiments, we've utilized a word cloud of the review texts that highlights the most frequently used words in Figure 2 and differentiates frequency using text font size. Common words include "good," "delicious," and "burger." Words such as "delicious" and "great" appear more frequently, which

Figure 2

are positive words that generally suggest a high satisfaction of the restaurant. This indicates that even within the reviews, users leave positive comments, which further strengthens our understanding of Figure 1 which tells us that users generally leave high ratings for a restaurant.

In Figure 3, we've used another Bar Graph to represent review length by rating, and we found that there is a clear trend between the length of reviews and their ratings. Looking at the different bars, which represents the average length of reviews with the rating, lower ratings tend to have longer reviews, suggesting that users provide more detailed feedback when they are dissatisfied. Combining this with our knowledge from Figure 3, although reviews are generally longer for lower rated restaurants, the prevalence of positive words are more dominant as suggested in the Word Cloud.

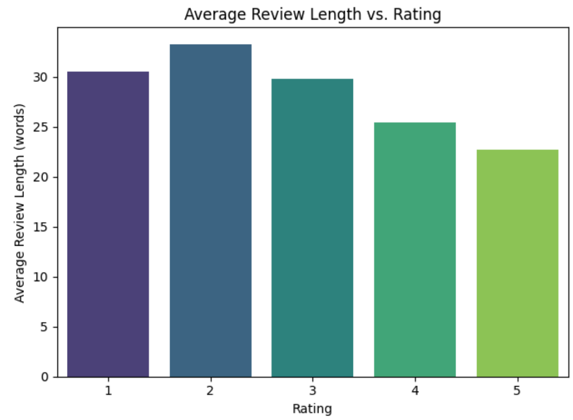

Figure 3

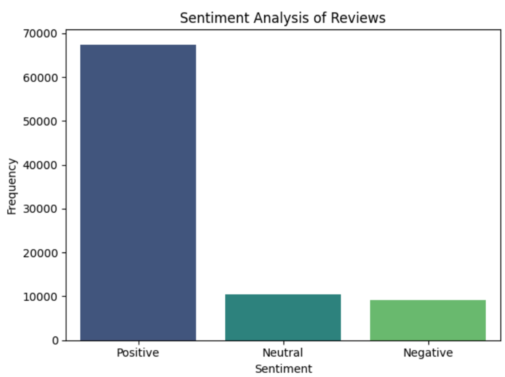

Figure 4

In observing the content of the reviews, we decided to use the TextBlob library to perform a sentiment analysis on the reviews. The TextBlob library performs sentiment analysis by calculating the polarity of the text, which ranges from -1 (very negative) to +1 (very positive). We visualized the results in Figure 4, a bar graph that showcases the frequency of positive, neutral, and negative sentiments within the reviews. The results show a higher number of positive sentiments compared to negative ones, aligning with the overall high average rating discovered from the previous Figures.

2 Predictive Task:

For our predictive task, if given a user and their respective previous history/reviews, we will predict/recommend 3 restaurants that they are most likely to enjoy. In the edgecase where a given user is not in our training set, we will recommend the top 3 most popular restaurants. Specifically, we plan to use each entry's `user_id` to identify a specific user, `business_id` for each restaurant, and the corresponding `rating`.

2.1 Data Processing

For processing the data, we simply extracted the necessary parts of the `json` file:

```
# l is the stored json
for d in l["train"]:
    train_dataset.append(d)

for d in l["val"]:
    val_dataset.append(d)

for d in l["test"]:
    test_dataset.append(d)
```

Then we created `users`, `business`, and `interactions` sets:

```
users = {}
business = {}
interactions = []

for d in train_dataset:
    if d["user_id"] not in users.keys(): users[d["user_id"]] = len(users)
    if d["business_id"] not in business.keys(): business[d["business_id"]] =
len(business)
    interactions.append((d["user_id"], d["business_id"], d["rating"]))

val_interactions = []
for d in val_dataset:
    if d["user_id"] not in users.keys(): users[d["user_id"]] = len(users)
    if d["business_id"] not in business.keys(): business[d["business_id"]] =
len(business)
    val_interactions.append((d["user_id"], d["business_id"], d["rating"]))

test_interactions = []
for d in test_dataset:
    if d["user_id"] not in users.keys(): users[d["user_id"]] = len(users)
    if d["business_id"] not in business.keys(): business[d["business_id"]] =
len(business)
    test_interactions.append((d["user_id"], d["business_id"], d["rating"]))
```

The `interactions` were then shuffled:

```
random.shuffle(interactions)
```

2.2 Evaluation

As for a baseline to compare our model with, we are using `Surprise`'s `SVD` function which is a built-in latent factor model. This baseline is found under `surpriseModel.ipynb`. Our model is a custom Latent Factor Model implemented via `PyTorch`, found under `brentford.ipynb`. In terms of evaluation between the two models and assessment of our model's validity, we used:

- **Mean Reciprocal Rank (MRR):** which measures how well our most recommended item ranks among the given label. Where $rank_u(i_u)$ is the label position of the first item in the predicted set (i.e. the most recommended restaurant):

$$MRR = \frac{1}{|U|}\sum_{u \in U}\frac{1}{rank_u(i_u)}$$

- **Precision@K:** where K = 3, which measures the proportion of how many of the top 3 predicted values are relevant:

$$precision@K = \frac{1}{|U|}\sum_{u \in U}\frac{\left|\left\{i \in I_u \mid rank_u(i) \le K\right\}\right|}{K},\ K = 3$$

- **Recall@K:** where K = 3, which measures the proportion of all relevant values that are in the top 3 predicted values:

$$recall@K = \frac{1}{|U|}\sum_{u \in U}\frac{\left|\left\{i \in I_u \mid rank_u(i) \le K\right\}\right|}{\left|I_u\right|},\ K = 3$$

- **Normalized Discount Cumulative Gain (NDCG):** where K = 3, which starts with the *Discounted Cumulative Gain (DCG)*, which counts how many relevant items are in the top 3 results (*Cumulative Gain: (CG)*) and *discounts* lower ranked items' contribution to the score. The DCG is then *normalized* by dividing it by the optimal DCG (*Ideal Discounted Cumulative Gain* (*IDCG*))
    - $NDCG@K = \frac{DCG@K}{IDCG@K},\ K = 3$
        - $DCG@K = \sum_{i \in \{i \mid rank_u(i) \le K\}}\frac{y_{u,i}}{log_2(rank_u(i) + 1)}$

- **Mean-Squared Error (MSE):** a metric used to calculate error, notably useful as it penalizes larger errors by squaring it. It is found by:
    - $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - X_i \cdot \theta)^2$

3 Model:

For our predictive task, we incorporated a latent factor model with the intention of capturing a relationship between user and item properties thereby recommending the items for a given user based on predictions built from these learned relationships. Other options that were considered include Sparse Linear Models [4], Bayesian Personalized Ranking [3] and using a similarity measurement for prediction.

A model-based approach was chosen in favor of similarity measurements due to the large data size which meant that a rich understanding of the variability of user interactions was needed. This was also the reason why Sparse Linear Model was not chosen given its user-free nature.

A latent factor model can be expressed by the following equation:

$$p(u, i) = \alpha + \beta_u + \beta_i \quad (1)$$

Where u, i are users and items respectively and β_u and β_i are values associated to user (u) and item (i) respectively.

Here we chose to test a latent factor model built on this equation and another model built on the following equation, as per Salakhutdinov & Minh [2]:

$$p(u, i) = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i \quad (2)$$

Where γ_u and γ_i are feature vectors for user(u) and item(i) respectively.

The purpose behind the choice of latent factor model is to learn the variations between users' opinions of items–in this case restaurants–without the need to incorporate too many features. This is also supported by the abundance of interactions found in the dataset.

The aim of using 2 different equations for the same type of model is to assess the role of model complexity when it comes to the given predictive task.

For the loss function, both models incorporated Mean Square Error loss as well as a regularizer to help deal with the overfitting thus generating the following loss functions for (1):

$$L(u, i) = \sum_{u,i}(p(u, i) - r_{u,i})^2 + \lambda(\sum_u (\beta_u)^2 + \sum_i (\beta_i)^2) \quad (1.1)$$

And the following for (2):

$$L(u, i) = \sum_{u,i}(p(u, i) - r_{u,i})^2 + \lambda(\sum_u (\beta_u)^2 + \sum_i (\beta_i)^2 + \|\gamma_u\|^2 + \|\gamma_i\|^2) \quad (2.1)$$

Where $r_{u,i}$ is the ground truth rating of user (u) with item (i) λ is a hyperparameter

When it comes to implementation, equation (1) was implemented using `Pytorch` such that the equation was optimized via Stochastic Gradient Descent. For (2), a scikit-surprise library was used with the hyperparameters tuned for performance. The library also used Stochastic Gradient Descent for its optimization. All parameters were randomly initialized via a normal distribution.

The following hyperparameters were chosen for (1):

- Learning rate for all parameters: 0.005
- Regularization terms: 0.5
- Number of epochs: 5
- Weight decay: 0.0001

The following hyperparameters were chosen for (2); Grid search was implemented to ensure that the selection provided an optimal performance:

- Learning rate for all parameters: 0.003
- Regularization terms:
    - For β_u: 0.01
    - For β_i: 2.5
    - For γ_u: 0.01
    - For γ_i: 0.06
- Number of epochs: 30

4 Literature:

Our model is trained on `filter_all_t.json` from Google Restaurants, a subset of Google Local Restaurants from this class's selection of datasets, where the images are highly correlated with sentences. Much of our attention was to recommend restaurants based on the `business_id`, `user_id`, and `rating` of an entry. When looking at the metadata, though, we found that text reviews provide an opportunity for sentiment analysis as we have explored earlier with TextBlob. Shin et al. [5] perform a similar process of text mining on restaurant reviews on Google Maps written in Korean to better understand the quality of service, price, and the food itself. Yang et al. [6] extend beyond the notion of "negative" and "positive" to analyze the semantics behind words and how they contextualize user preferences for certain foods from the Yelp Open Dataset. Their use of aspect-based sentiment analysis (ABSA) is a novel approach to enrich latent factors with interactions between customers and restaurants.

Systems resembling those of Netflix and Amazon leverage collaborative filtering by grouping user/user and item/item similarity to match the preferences of a user in question with those of existing users. Khadka et al. [7] highlights that the algorithm relies on ratings, item choice, price ranges, along with other heuristics to develop an understanding of user preference and similarity. It is important to note that collaborative filtering works well on a system like Netflix given the high volume of activity to build a diverse network of tastes among users. The constant flow of reviews is necessary when handling new users and items to accurately recommend unfamiliar and, therefore, unsimilar variables. When it comes to the actual similarity metrics, Amazon uses the Cosine similarity

$$Sim(u, v) = \frac{\Sigma_{i \in I_u \cap I_v} R_{u,i} R_{v,i}}{\sqrt{\Sigma_{i \in I_u \cap I_v} {R_{u,i}}^2 \, \Sigma_{i \in I_u \cap I_v} {R_{v,i}}^2}}$$

for the many undefined ratings, but there are also measures like Pearson Correlation Coefficient

$$Sim(u, v) = \frac{\Sigma_{i \in I_u \cap I_v} (R_{u,i} - \overline{R_u})(R_{v,i} - \overline{R_v})}{\sqrt{\Sigma_{i \in I_u \cap I_v} (R_{u,i} - \overline{R_u})^2 \, \Sigma_{i \in I_u \cap I_v} (R_{v,i} - \overline{R_v})^2}}$$

when we must consider the personalized rating scales of different users.

Content-based filtering is also used to tackle personalized recommendations for users that have a previous history of reviews. Within the context of restaurant recommendations, Melese [8] explains that preferences consider attributes specific to users like their I.D., ratings, and descriptions for food. Term frequency/inverse document frequency (TF-IDF) scores weigh food attributes relative to their frequency in a dataset and capture what food text keywords are important. The tasks to consider are:

$$TF_{xy} = \frac{f_{xy}}{max(f_{ij})} \qquad \text{and} \qquad IDF_x = log\frac{X}{X_i}$$

Where we try to measure the number of times a food keyword f_{xy} appears in a dataset and also the quantify the number of datasets X in relation to X_i, the datasets that contain our keyword *i*.

Our use of Surprise's SVD algorithm is quite similar to the matrix factorization logic from Yang, Li, Jang, and Kim with the Yelp restaurant reviews as it more adequately handles the sparsity in our data. Thus, it becomes clear that collaborative and content-based filtering would not bode as well in this case as there are simply not enough entries within this dataset that would make said approach reliable.

5 Conclusion

Our study aimed to predict the top 3 restaurants that a user would most enjoy based on their reviews using a latent factor model implemented as outlined in **3 Model (equation 1)** via `PyTorch`. This was then compared to a latent factor model using `Surprise`'s `SVD` function. The results are shown below respectively.

Metric	`PyTorch` **Model**	`Surprise SVD` **Model**
MRR	0.9494977574977576	0.9544144144144145
precision@3	0.7966666666666425	0.7966666666666425
recall@3	0.9159855047592548	0.9159855047592548
NDCG	0.9416	0.9507802138805759
MSE	1.9404861194056433	0.6769716283799662

From this, we observe that the `Surprise SVD` model outperforms our `PyTorch` model in **NDCG**, **MRR**, and **MSE** while both models performed identically for **precision@3** and **recall@3**. The higher **MRR** score for the `Surprise` model tells us that it does a better job at recommending a more relevant top restaurant than our model, but only slightly so. Similarly, the higher **NDCG** score results favor the `Surprise` model, which suggests better quality in its ranked recommendations. Furthermore, the `Surprise SVD`'s significantly lower **MSE** score suggests that it outperforms our `PyTorch` model with rating predictions of the restaurant themselves, but not necessarily in what would be the top 3 restaurants for a user. This could be the result of the feature vectors of the `SVD` implementation. The `PyTorch` model had achieved identical results for **precision@3** and **recall@3**, which more importantly indicates that both models performed similarly at gathering relevant data for the top 3 predictions.

With β representing the variability of user interaction with items and γ being the feature vectors of users and items, we saw that increasing the complexity and learning the feature vectors did not increase performance by a significant amount (as both models performed similarly). Although the `SVD` model performed better in terms of **MSE**, we are more focused on how they measured up in the relevancy of the 3 recommendations, not how good the rating for each independent restaurant was.

In conclusion, with our simpler model, we were able to achieve similar results to a more complicated model and with less fine tuning of hyperparameters, marking our proposed model a success.

6 References

Our GitHub Repository: https://github.com/AZA-2003/CSE158-A2

[1] Personalized Showcases: Generating Multi-Modal Explanations for Recommendations
An Yan, Zhankui He, Jiacheng Li, Tianyang Zhang, Julian McAuley
arXiv:2207.00422, 2022. Pdf.

[2] Probabilistic Matrix Factorization
Ruslan Salakhutdinov and Andriy Minh. 2007. Pdf.

[3] BPR: Bayesian personalized ranking from implicit feedback.
Rendle, Ste en, Freudenthaler, Christoph, Gantner, Zeno, and Schmidt-Thieme, Lars. 2012. *arXiv:1205.2618.* Pdf.

[4] Slim: Sparse linear methods for top-n recommender systems
Ning, Xia, and Karypis, George. 2011. Pdf.

[5] Analysis on Review Data of Restaurants in Google Maps through Text Mining: Focusing on Sentiment Analysis
Shin, Bee, Ryu, Sohee, Kim, Yongjun, Kim, Dongwhan. 2022. Pdf.

[6] Deep learning mechanism and big data in hospitality and tourism: Developing personalized restaurant recommendation model to customer decision-making
Yang, Sigeon, Li, Qinglong, Jang, Dongsoo, Kim, Jaekyeong. 2024. Pdf.

[7] Restaurant Recommendation System Using User Based Collaborative Filtering
Khadka, Salu, Chaise, Pragya Shrestha, Shrestha, Sujin, and Maharjan, Satya Bahadur. 2020. Pdf.

[8] Food and Restaurant Recommendation System Using Hybrid Filtering Mechanism
Melese, Amanuel. 2021. Pdf.