COMP4332 Project 3 Report
Recommendation System

Group 14

1 Introduction

The rise of the internet and the emergence of vast amounts of data have brought about unprecedented access and control for individuals to choose how to spend their time and money. But the sudden availability of countless options poses the challenge of selecting the best possible choice.

By using a recommendation system, information regarding a user and a set of prospective products can be used to suggest relevant items. This task is of great importance for businesses with massive amounts of data. Consequently, large companies such as Google, Netflix, Spotify, Amazon, and Yelp have all developed their own models and systems. Some of them, including Netflix and Yelp, have even held public challenges and competitions to develop the most robust systems.

In this paper, we participate in the Yelp Recommendation challenge, which aims to provide ratings to businesses based on different user and business data. Our overall performance will be measured on RMSE, the root mean square error, between our predictions and the actual ratings.

2 Data Observation

There are three major parts of the data: review, user, and business. The review file consists of the stars given by the users for the businesses. Both users and businesses are given in a json format file consisting of different information and identified by their id.

The business.json file provides a wide variety of information regarding different businesses. For each business, as identified via their business id, the following information were provided: address, latitude, longitude, attributes of the business, business category or type of business, opening hours, name, postal code, review count, average stars received, city and state of operation, and postal code.

For the user.json file, the following information were provided: average stars given, the number of cute, hot, more, list, writer, photo and plain compliments received, the number of fans, and the number of useful, funny, and cool votes sent.

It is observed that the training data consists of 100,000 entries of review, while the testing and validation consists of 10,000 entries. The information provided in the training.csv has the user id of a certain user, the business id of the business being reviewed, and the corresponding number of stars provided by the user for the business. In all, the information provided was a mix of continuous, categorical and textual data. Due to this, extensive pre-processing was required to transform the data to usable features that can be passed on to the model. The exact details of the pre-processing and feature engineering methods will be detailed in the next section.

3 Data Pre-processing and Feature Engineering

Since the data about users and business were separated into different json files. The first thing to do was to merge both information into a single data frame. Columns originating from user.json file were appended with **user** at the start of their names while **item** was appended to the names of columns originating from the business.json file.

We then separated the data (columns) between continuous and categorical data. The following columns are the list of continuous features:

- user_average_stars
- user_compliment_cool
- user_compliment_cute
- user_compliment_funny
- user_compliment_hot
- user_compliment_list
- user_compliment_more
- user_compliment_note
- user_compliment_photos
- user_compliment_plain
- user_compliment_profile
- user_compliment_writer
- user_cool
- user_fans
- user_funny
- user_review_count
- user_useful
- item_latitude
- item_longitude
- item_review_count
- item_stars

As the data have different scales, it was important to perform scaling to the features above. Standard scaling was used, wherein each feature is made to be normally distributed.

The following columns have categorical data, which were found to be relevant for the task:

- item_is_open
- item_attributes
- item_city
- item_hours
- item_categories
- item_postal_code

Further data extraction was performed on the **item_attributes** column as each item in the column had more features or attributes about the corresponding business. The following are the additional features used:

- NoiseLevel
- RestaurantsAttire
- RestaurantsTakeOut
- RestaurantsReservations
- RestaurantsDelivery
- Alcohol
- RestaurantsPriceRange2
- BikeParking
- HappyHour
- OutdoorSeating
- RestaurantsGoodForGroups
- HasTV
- Caters
- GoodForKids
- BusinessAcceptsCreditCards
- WiFi
- GoodForDancing
- Smoking
- RestaurantsTableService
- Corkage
- CoatCheck
- BYOB

A vocabulary of the possible values of each feature listed above was then created, and a corresponding id value was mapped to each value's position in the feature's vocabulary. Note that not every business has each of the features listed above. As a result, a NaN value is given to the missing feature of a restaurant, which maps to the 0 index in a feature's vocabulary. The same was done for the other categorical features listed above.

4 Training Environment

As additional data were used during experimentation, Intel's DevCloud service was utilized to process the full dataset that consists of several GBs. Expansion of data is made convenient by using free high quality CPUs available in Intel's cluster.

In addition, Google Colab is mainly used for training the model, as it provides high memory, with roughly 12GB of available RAM, and good quality CPUs and TPUs. Although no possibilities of using GPU hardware acceleration arose for this project, Google Colab still provides a suitable training environment.

5 Experiments and Results

5.1 Data Addition

In one experiment, more data downloaded from the full Yelp dataset were provided to the model. Additional 100,000, 200,000, and 400,000 data for training and their respective user and business data were added as input to the model. However, the performance of the model is observed to be the same even with the additional training data provided.

In addition, data on compliments received by each user for their reviews, photos, and status are used as additional input for the deep features. After concatenation with the embedding layers, both 1, 2, and 3 Fully-Connected Layers having 256 and 128 hidden nodes were used. However, this addition of compliment data did not increase the performance of the model significantly, as it yielded RMSE between 1.033 and 1.034.

5.2 Final Model

The final model is based on the Deep and Wide Model released by Google for Item Recommendation. After several rounds of experimentation, the performance of this model was found to be significantly better than models such as Neuro-Collaborative Filtering. This model has two main sections, the **deep** part of the model and the **wide** part of the model.

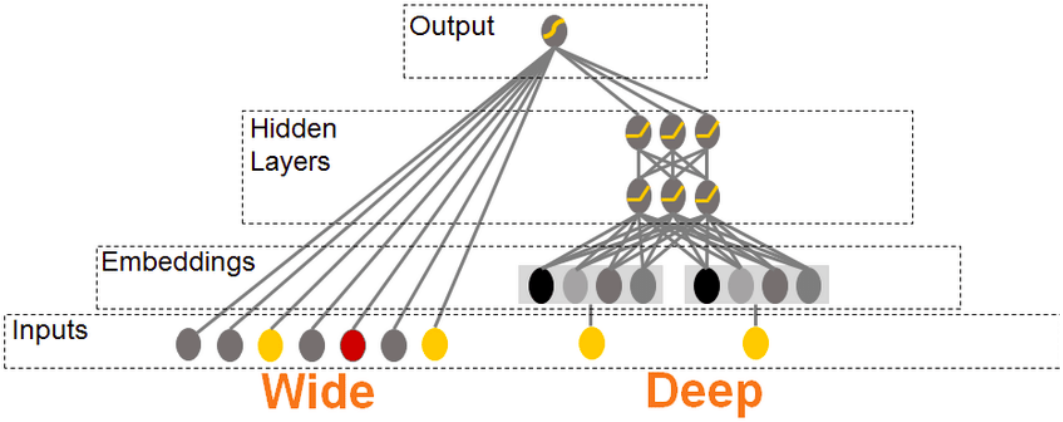

Figure 1: Deep and Wide Model Architecture

The **deep** part of the model can be further divided into continuous features and categorical features, as mentioned in Section 3. Most of the categorical features are extracted from **item_attributes**, which contains details about the item. As for the continuous data, a scaling is performed in order to normalize the values.

For each categorical feature, we add an embedding layer of size **10**. These embeddings are then concatenated with the values from the continuous features. Afterwards, we feed the embedded values into a Dense layer of size **1024**, with ReLU as the activation function and a dropout rate of **0.2**.

The **wide** part utilizes the values inside **item_categories**. It first extracts all categories that are labeled for each item and filters the top 500 most common. Items usually have more than one category, so we also find possible combinations of 2 categories, 3 categories, and 4 categories and filter out the top 70, 20, and 10 respectively.

Next, the wide features are obtained by generating an array of binary outputs, whose values corresponds to the 500 selected category and the category combinations by assigning 1 if the item belongs to the category or category combination and 0 otherwise.

Lastly, we concatenate the outputs from **deep** and **wide** parts and output it to a Dense layer of size **1**, which is the rating.

6 Conclusion

As the final result, we managed to come up with a recommendation system model that matches the ratings of items with the users. The Deep and Wide Model is the right choice, seeing from the nature of the problem itself and the features of the dataset. Furthermore, increasing the amount of features included in the **deep** part of the model by utilizing data from **item_attributes** gave a substantial improvement of the model. We successfully reached a value of 1.0293240753474389 on the validation data using the RMSE measure.