Team #138: Steve Zheng, Cuong Pham, Jonathan Mogan, Trang Doan

Project Title: Yelp’s Recommender System

I. Introduction:

In today’s digital landscape, where users are presented with an overwhelming array of options, personalized recommendation systems are vital for enhancing user satisfaction and engagement. Traditional recommendation methods often rely primarily on general user-item interactions, which can overlook the finer details of personal preferences that significantly shape user experiences. This is particularly evident in the restaurant industry, where preferences for dining experiences are influenced by factors such as food quality, service timeliness, and specific tastes. To address these nuances, we introduce a network-driven restaurant recommendation tool that harnesses the rich data within Yelp's dataset. This tool combines user-item collaborative filtering with sophisticated sentiment analysis to generate recommendations that reflect both past behaviors and detailed individual preferences. By incorporating aspect-based sentiments from user reviews, our model identifies preferences toward food quality, timeliness of service, and specific taste profiles. This approach allows us to offer dining recommendations that are not only relevant but also tailored to align with users’ personal expectations for a dining experience.

II. Literature Survey:

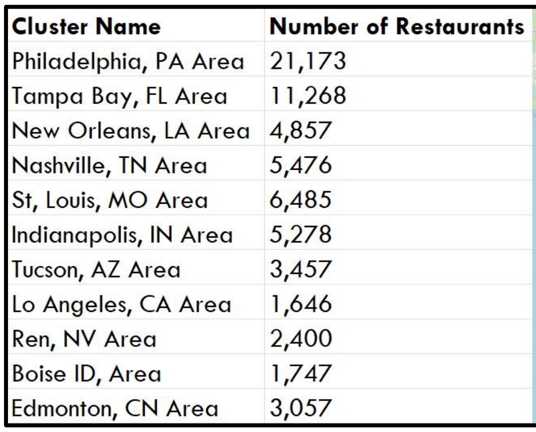
Yelp's current model deploys 2 sets of features in its model: (a) Interaction features (b) Content- based features with the key feature: text-based similarity. Reviews are encoded and aggregated at the business level, then further aggregated at the user level by linking users to businesses they have reviewed. Cosine similarity is used to measure the relationship between user-level and business-level embeddings, helping enhance the recommendation system’s accuracy by comparing these aggregate representations.[2] Despite its costly join operation, the old matrix factorization technique which decomposes the large user-item interaction matrix (where users rate restaurants) into two smaller matrices: one representing user preferences and the other representing item (restaurant) characteristics is still used as an input for the hybrid model.[1] Yelp uses Universal Sentence Encoder (USE) to transform user reviews/texts into a fixed-length vector representation that captures the semantic meaning of the sentences [10]. As most NLP models perform better when training with domain-specific text, Yelp adapted the pre-train USE model and fine-tuned it specifically for reviews. [9] Although USE can be used for sentiment analysis, user reviews often combine both satisfaction and dissatisfaction, which can make the overall sentiment misleading. A variation to the Yelp hybrid recommender model, the HybridBERT4Rec, leverages BERT to account for changing user interests over time by ingesting temporal data.[8] It intakes sequences of users who rated a target item and series of items that a target user interacted with, and generates a user and item profile. These two are combined to predict the likelihood of the user buying the item.

Aspect Based Sentiment Analysis utilizing Bidirectional Encoder Representations from Transformers (BERT), the chosen methodology for classifying Yelp Reviews as having either a positive, neutral, or negative sentiment, is a methodology which works by pretraining a deep network of bidirectional representations of data which, when combined with fine-tuning of a final output layer, can yield results specific to the task at hand, in this case classifying reviews based on the sentiment of the text [12]. Currently, Yelp utilizes a Universal Sentence Encoder (USE) model, which does capture user sentiment in determining whether reviews are positive or negative in nature. Tsukagoshi et all’s “Comparison and Combination of Sentence Embeddings Derived from Different Supervision Signals” [3] highlights that USE models, pre-trained on Natural Language Inference datasets, are highly effective at generating accurate and fine-tuned sentence embeddings. However, Aspect-Based Sentiment Analysis allows for the ability to fine-tune the output for sentiment based on specifically chosen aspects. BERT, when compared to other Machine Learning, Statistical, and Lexicon-Based models, can be shown to perform as well as or better than the aforementioned models. Additionally, with respect to Aspect-Based Sentiment Analysis, BERT has the added benefit of being able to select for specific portions of text which the algorithm should focus most on when determining sentiment without the specification for which aspects exist in the corpus beforehand [4]. BERT also has the benefit of pre-training on large bodies of text specific to the use case, making it easier to tailor the model to the application it is being used for. In Hoang et all's "Aspect-Based Sentiment Analysis Using BERT", when looking at the ability to make determinization of sentiment with respect to training on both in-domain and out-of-domain corpuses, in this case utilizing laptop reviews and food reviews together and separately to train an ABSA model, accuracy of classification ranged from 81.2% for sentiment generated for restaurant reviews when trained only on laptop reviews, all the way to 89.8% for sentiment generated utilizing a corpus of both laptop reviews and food reviews [5]. This demonstrates the model's ability to be adapted for various uses and applications, especially when training on a domain of text that is relevant to the data which will eventually be classified. Because BERT-based models are computationally expensive, we implement mixed precision quantization which uses a lower precision of 8 bits or 16 bits to reduce computation complexity and maintain the model's accuracy.[11] Another key factor in restaurant recommendation systems is location and distance measurement. Services like Google Maps API can be leveraged to pull driving distance between two locations. However, other related works have shown that straight-line distance can adequately substitute driving distance. An article from the National Institute of Health studied distance between homes and hospitals and found a very high correlation ($r2 > .9$) between straight-line and driving distance. [6] A Journal of Physics article calculated distance between temporary waste collection and final processing sites using the Haversine distance formula, which uses a pair of lat-long coordinates to output the straight-line distance accounting for Earth's curvature. [7]

III. Methods/Approaches:

Combining sentiment prediction with a restaurant recommender system offers a promising enhancement over current state-of-the-art methods by creating a more nuanced and responsive user experience. Traditional recommendation algorithms, which typically rely on collaborative filtering or content-based approaches, mainly account for past ratings and general preferences but often miss the critical context of user sentiment. By incorporating sentiment prediction, the system can interpret not only what users' rate but also how they feel about specific aspects—such as service quality, ambiance, or food options. This approach allows recommendations to adjust dynamically in real time, aligning more closely with the user's current preferences for elements like a top tier food taste or quick, efficient service. By tapping into both sentiment and ratings, the system can deliver more personalized restaurant suggestions that better match users' emotional expectations, making it a more sophisticated and user-centered tool than conventional recommendation models.

We used a Yelp dataset of around 7 million reviews from 2 million users across 150,000 businesses. For our restaurant recommendation system, we filtered it to focus solely on restaurant businesses and reviews. We leveraged the "categories" feature provided for each business, which was a list of strings tagged for that business. We first filtered for businesses that contain the "food" category. Then, we extracted the top 20 most common categories within this subset. The most common category associated with "food" was, unsurprisingly "restaurants", followed by "coffee & tea" and "fast food". Finally, we filtered for any business with a category containing "food" OR any of these top 20 associated categories and used this as our list of restaurant businesses. Our filtered dataset contains 67K businesses and 5.2M reviews, providing relevant and meaningful training data for our system. To speed up the Recommendation algorithm and to ensure that recommended restaurants were within a reasonable distance of the user, all restaurants were grouped according to their latitudinal and longitudinal coordinates. Based on figure 1 below, we can see that the restaurant reviews exist within 11 densely populated clusters, near major cities spread out across both the United States and Canada.



Cluster Name	Number of Restaurants
Philadelphia, PA Area	21,173
Tampa Bay, FL Area	11,268
New Orleans, LA Area	4,857
Nashville, TN Area	5,476
St, Louis, MO Area	6,485
Indianapolis, IN Area	5,278
Tucson, AZ Area	3,457
Lo Angeles, CA Area	1,646
Ren, NV Area	2,400
Boise ID, Area	1,747
Edmonton, CN Area	3,057

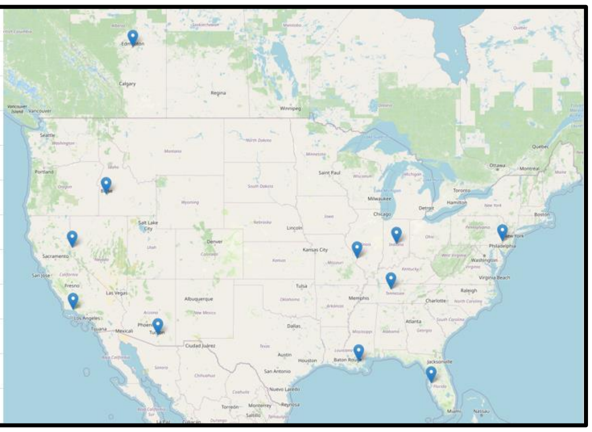


Figure 1: Restaurant Clusters

The above cluster centers were calculated using a K-means algorithm. First, visualizing the restaurants on a map to determine where clusters are likely to reside, then initializing a good starting point for each of the cluster centers to ensure that the results are representative of the data. These cluster centers were then saved and are now utilized by the application, querying the user's current longitude and latitude from their browser, then assigning them to the closest cluster utilizing Haversine Distance. This ensures that the results of the application are only for those restaurants that are located in the same metropolitan area in the cluster closest to the user's current location. Additionally, this has the added benefit of cutting the number of restaurants that the Matrix Factorization algorithm must use, reducing the amount of wasted computation spent comparing the user's preferences with restaurants that aren't a reasonable distance from the user. The clustering algorithm will then return a choice of the 50 top-rated restaurants in each cluster for the user to choose from. These restaurants are returned to the user in the form of a dropdown list which they are then able to use in choosing which restaurants they have enjoyed before. If the user is new to the system, we will add their inputs here into the user-item matrix to avoid cold start problems.

The rating of user u_a on item i_j in the dataset is then predicted as follows:

$$pr_{aj} = \beta * prmf_{aj} + (1-\beta) * prsent_{aj}$$

Below we discuss the approaches used to derive $prmf_{aj}$ and $prsent_{aj}$.

Matrix Factorization models

The two matrix factorization models used are basic matrix factorization and a variation of matrix factorization using neural network.

The process begins by generating user and item embeddings. The user-item interaction matrix is factorized into two lower-dimensional matrices: one representing user embeddings and the other representing item embeddings. These embeddings capture latent features, encoding user preferences and item characteristics in a shared feature space. The below equation summarizes this:

$$p_a = useremb(u_a) \in \mathbb{R}^{embsize}, \quad q_j = itememb(i_j) \in \mathbb{R}^{embsize}$$

The basic matrix factorization model computes the dot product of the user and item embeddings:

$$prmf_{aj} = p_a{}^T q_j = \sum_{n=1}^{embsize} p_{a,n} \cdot q_{j,n}$$

A neural network model builds on top of the basic model through two additional layers:

(1) Fully Connected Layer: This layer serves as the primary computational unit in the network. It takes the concatenated latent embeddings of users and items (learned from matrix factorization) as input and maps them to a new feature space. Each neuron in the fully connected layer is linked to every input feature, allowing the model to learn complex interactions between user and item embeddings. This layer typically applies a non-linear activation function (ReLU) to introduce non-linearity and enhance its capacity to model complex relationships.
(2) Dropout Layer: To prevent overfitting, a dropout layer is applied after the fully connected layer. This regularization technique randomly deactivates a specified fraction of neurons during training, encouraging the network to learn robust and generalized patterns. The dropout rate (e.g., 0.2 or 0.5) controls the proportion of neurons dropped.

This neural network structure allows the model to learn non-linear relationships between user and item embeddings, going beyond the simple dot product of traditional matrix factorization models. The structure is summarized below:

$$prmf_{aj} = W_2 ReLU(W_1[p_u; q_v] + b_1) + b_2 \ where \ W_2 \in \mathbb{R}^{1xn_{hidden}}, b_2 \in \mathbb{R}$$

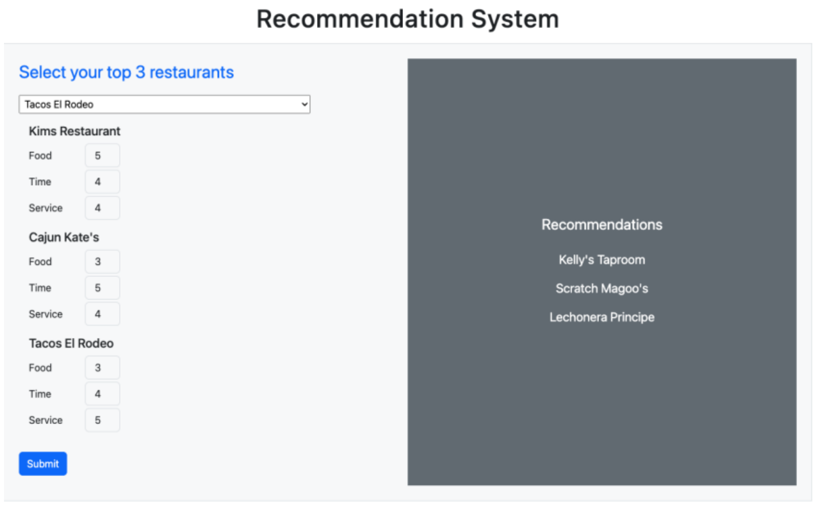
We optimized the neural network parameters using PyTorch and the Adam optimizer to minimize the error between actual and predicted ratings. A Mean Squared Error (MSE) loss function, combined with regularization, helps prevent overfitting.

Sentiment models

To compute $prsent_{aj}$, the process begins by using a pre-trained BERT model to classify each review into 1-3 score scale for the aspects of Food, Service, and Time. The review rating scale is from 1 to 5 so we transform predicted sentiment score to the same scale. For a given user u_a, all items they have rated are identified, along with a single aggregated sentiment score for each item. In the second step, for a given item i_j, all users who have rated both i_j and item from the first step (i_k are identified, along with their aggregated sentiment scores for these items. Using these filtered sets of items and users, the similarity between the active user u_a and other users u_b is computed using the cosine similarity metric. This similarity is then used to select the K-nearest neighbors. Finally, the predicted sentiment-based rating $pr_{sent_{aj}}$ for user u_a and item i_j is calculated by weighing the sentiment ratings of the K-nearest neighbors based on their similarity scores. See citation [13] for the reference paper.

The selection of the parameter β plays a crucial role in balancing the contributions of explicit ratings and sentiment-based feedback in the final prediction. To identify the optimal value for β, we employ grid search, systematically evaluating a range of potential values. This approach involves fine-tuning β by testing its performance across the range between 0 and 1 and selecting the value that minimizes the error metrics MSE on the validation dataset. The optimal value is 0.6. Based on the predicted rating, the model then will return the top 5 suggestions to the user.

The frontend is a clean, straightforward interface for the user to submit 3 existing restaurant preferences to the model and then receive 3 other recommended restaurants based on the input. For each of the 3 restaurant inputs, the user also rates the food, time, and service aspect from 1-5. This information gets used in the sentiment analysis part of the model. See Figure 2 for an example:


Recommendation System

Select your top 3 restaurants

Tacos El Rodeo

Kims Restaurant

Food	5
Time	4
Service	4

Cajun Kate's

Food	3
Time	5
Service	4

Tacos El Rodeo

Food	3
Time	4
Service	5

Submit

Recommendations

Kelly's Taproom

Scratch Magoo's

Lechonera Principe

Figure 2: Screenshot of user interface

IV. Experiments / Evaluation

The testbed for evaluating the recommender system is built on a dataset containing user-item interaction data, split into training and testing sets to assess performance. The system includes collaborative filtering using matrix factorization combined with sentiment prediction based on three chosen aspects. It is evaluated on the Mean Square Error metric. Experiments aim to answer key questions: How accurate and relevant are the recommendations? How well does the system rank items for users? Can it handle scalability, cold-start problems, and noisy data effectively? These tests ensure the system is robust, efficient, and impactful.

The first experiment aimed to assess the model ‘s performance for sentiment classification, focusing on ABSA with BERT to determine its effectiveness. Using a labeled corpus of Yelp reviews containing review text and sentiment labels (positive or negative), the model's performance was tested. Because the chosen model included a neutral category, the algorithm was adjusted to eliminate it, reassigning classifications to positive or negative sentiments based on probability. Each of the three aspects was individually tested on a sample of 100,000 reviews, achieving an average accuracy of 80-90% for both positive and negative sentiments. These results confirm the ABSA BERT model's effectiveness for sentiment classification in Yelp’s dataset. To evaluate the performance of matrix factorization and neural networks for embedding the user-review matrix, experiments were conducted using the same dataset of user-item interactions. For matrix factorization, techniques like alternating least squares (ALS) were implemented to generate embeddings based on latent factors. For the neural network, a deep learning model was trained to learn embeddings by optimizing a loss function that minimizes the error between predicted and actual ratings. The metrics used to compare the methods included the accuracy of rating predictions and the time taken for training and inference. These assessments ensured the chosen approach balanced predictive accuracy with computational efficiency, providing effective and scalable embeddings for the task. To validate recommendation accuracy, we performed out-of-sample testing experiments by removing specific users from the model dataset, deriving model predictions based on those users’ preferences, and finally comparing the predictions to the users’ actual ratings. We visualized the comparison of actual vs predicted using boxplots; see below for an example:

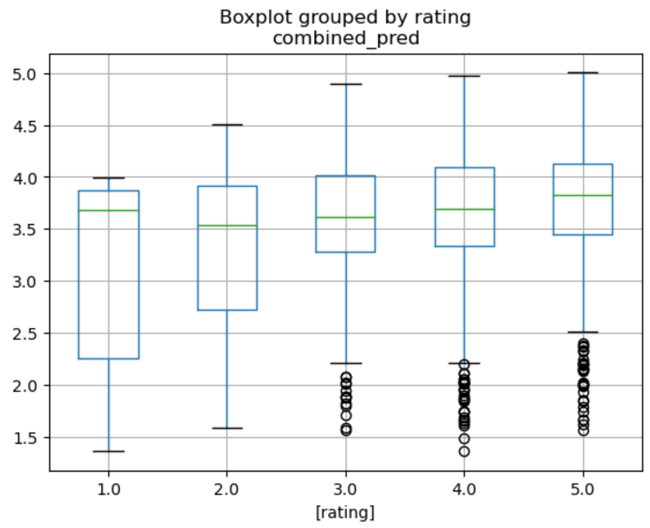


Figure 3: Distribution of Predicted Ratings Grouped by Actual Rating, for one user

The above boxplot shows the distribution of model predicted ratings, grouped by restaurants with the same actual rating for a given user (user_id = Xw7ZjaGfr0WNVt6s_5KZfA). In other words, the boxplot above the 1.0 on the x-axis shows the quartiles of model predictions on all restaurants that the user actually rated as “1.0”. If the model were perfect, we would see 5 points along the 45-degree diagonal from bottom left to top right; all restaurants that the user rated a 1.0 would also be predicted as 1.0, and so on for each actual rating. Based on the current boxplot, the model performance is mixed. The medians hover between 3.5 to 4 irrespective of actual ratings (though there is a slight increase in the median predicted rating for higher rated restaurants), suggesting that the model tends to predict a rating between 3.5 to 4 for any restaurant. On the other hand, the interquartile range is smaller for higher rated restaurants. This means that the model has less variance in its predictions for better restaurants, whereas for worse restaurants there is large variation below the median. Plots for other users tell similar stories, which we have dumped in the Appendix.

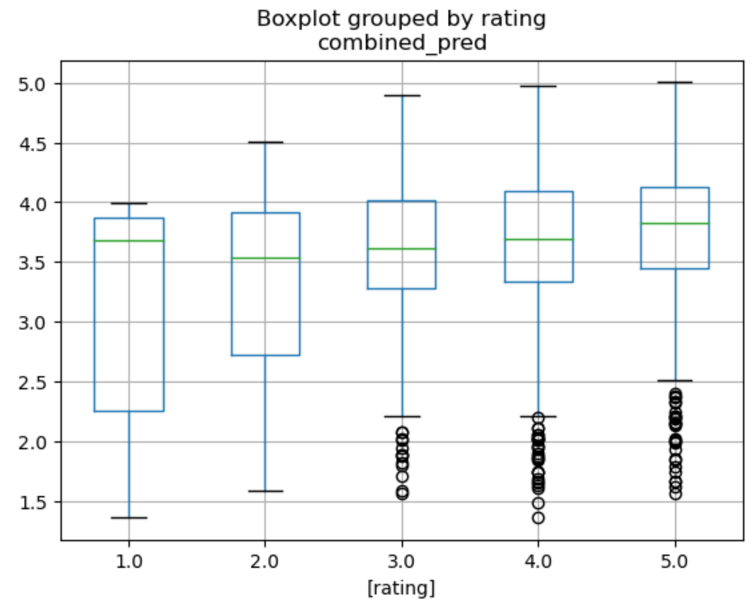
V. Conclusions and discussion:

There are a few limitations to the current app design. First, there is room to improve our matrix factorization implementation, to better balance prediction accuracy with practical runtime. We found that a neural network model with more hidden layers scored better than a classic matrix factorization model with only the user and restaurant embedding layers, but the best-performing NN model took ~4 minutes to output predictions in our testing. As such, our app uses a less accurate model, but provides more timely output (still ~1.5 minutes from input submission to output generation).

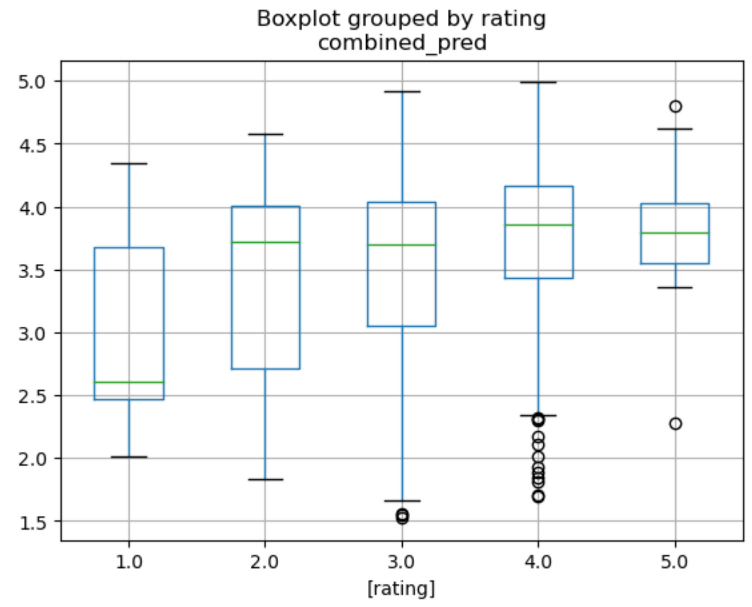
There is also a data limitation, namely that the Yelp dataset only contains restaurants in 11 clusters. Users are mapped to the closest cluster to their location, and restaurant recommendations are only from that cluster. Therefore, if a user is not based in any of the 11 clusters, the app recommendations may feel less relevant.

Throughout this project, all team members have contributed a similar number of efforts.

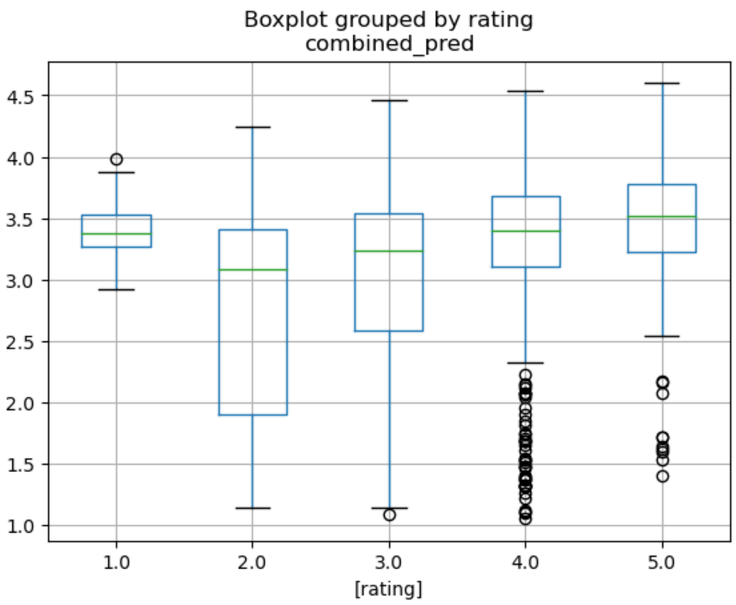
Appendix



Boxplot of Predicted Ratings by Actual Ratings, for user_id Xw7ZjaGfr0WNVt6s_5KZfA (Figure 3 in paper)



Boxplot of Predicted Ratings by Actual Ratings, for user_id zF10BKjK4Lz9U_8Yqw28ww



Boxplot of Predicted Ratings by Actual Ratings, for user_id bYENop4BuQepBjM1-BI3fA

Citations

1. Bashar, S., & Gillmor, A. (2018, May 7). Scaling collaborative filtering with PySpark. Yelp Engineering Blog. https://engineeringblog.yelp.com/2018/05/scaling-collaborative-filtering-with-pyspark.html
2. Rajagopalan, S. (2022, April 25). Beyond matrix factorization: Using hybrid features for user-business recommendations. Yelp Engineering Blog. https://engineeringblog.yelp.com/amp/2022/04/beyond-matrix-factorization-using-hybrid-features-for-user-business-recommendations.html

3. Tsukagoshi, H., Sasano, R., & Takeda, K. (2022). Comparison and Combination of Sentence Embeddings Derived from Different Supervision Signals (pp. 139–150). https://aclanthology.org/2022.starsem-1.12.pdf

4. Joyeux, D., Hasan, O., & Brunie, L. (n.d.). A Study and Comparison of Sentiment Analysis Methods for Reputation Evaluation. https://liris.cnrs.fr/Documents/Liris-6508.pdf

5. Hoang, M., Oskar, A., Bihorac, & Rouces, J. (n.d.). Aspect-Based Sentiment Analysis Using BERT. https://aclanthology.org/W19-6120.pdf

6. Boscoe, F. P., Henry, K. A., & Zdeb, M. S. (2012). A Nationwide Comparison of Driving Distance Versus Straight-Line Distance to Hospitals. The Professional geographer : the journal of the Association of American Geographers, 64(2), 10.1080/00330124.2011.583586. https://doi.org/10.1080/00330124.2011.583586

7. Azdy, R. A., & Darnis, F. (2020, April). Use of haversine formula in finding distance between temporary shelter and waste end processing sites. In Journal of Physics: Conference Series (Vol. 1500, No. 1, p.012104). IOP Publishing.

8. Channarong, C., Paosirikul, C., Maneeroj, S., & Takasu, A. (2022). HybridBERT4Rec: a hybrid (content based filtering and collaborative filtering) recommender system based on BERT. IEEE Access, 10, 56193-56206.

9. Yelp Content As Embeddings. (2023). Yelp.com. https://engineeringblog.yelp.com/2023/04/yelp-content as-embeddings.html

10. Cer, D., Yang, Y., Kong, S.-Y., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., & Kurzweil, R. (2018). Universal sentence encoder. ArXiv. https://arxiv.org/abs/1803.11175

11. Pandey, N. P., Nagel, M., Baalen, van, Huang, Y., Patel, C., & Blankevoort, T. (2023). A Practical Mixed Precision Algorithm for Post-Training Quantization. ArXiv.org. https://arxiv.org/abs/2302.05397

12. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the

North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 4171–4186. https://doi.org/10.18653/v1/N19-1423

13. Dang CN, Moreno-García MN, Prieta FDl. An Approach to Integrating Sentiment Analysis into Recommender Systems. Sensors. 2021; 21(16):5666. https://doi.org/10.3390/s21165666