From Ratings to Recommendations: Predictive Models for Personalized Local Business Experiences

December 3, 2024

I. Abstract

This study explores a predictive task to recommend highly-rated businesses based on user preferences and geographic constraints and aims to build a robust recommendation system utilizing a subset from the Google Local Data (2021) dataset that includes Google Maps reviews within a 5-mile radius range from the San Diego city center. Various collaborative filtering and advanced models, such as matrix factorization and behavior sequence transformers, were applied to predict user ratings for unseen businesses. The results of this study demonstrated the effectiveness of sequential and latent factor approaches in capturing user-item interactions, highlighting the challenges and trade-offs in real-world recommendation systems.

II. Literature Review

Widely used Internet and smart devices have become platforms in which various information and activity data including ratings, reviews, and descriptions can be collected. By analyzing a vast amount of data, research on filtering and providing reasonable recommendations customized to users' tastes or needs based on datasets similar to the one chosen in this study have rapidly increased over the past thirty years. The limitations of the existing models were continuously supplemented and developed.

A. Collaborative Filtering Models

B. Sarwar et al. (2001) proposed item-based collaborative filtering recommendation algorithms, providing better performance than the basic k-nearest neighbor approach and user-based algorithms on the MovieLens dataset, while providing better quality than the best available user-based algorithms. Specifically, it calculates item-item similarities and different techniques to generate recommendations. Y. Koren et al. (2009) proposed matrix factorization (MF) models that learn the general taste of a user by factorizing matrix over observed user-item preferences, which is superior to classic nearest-neighbor techniques for producing product recommendations in Netflix Prize competition.

B. Content-Based Models

J. Pazzani et al. (2007) discussed content-based recommendation systems that analyze the user-item matrix to discover relations between the different items and compute the list of recommendations. The key steps are (i) computing the similarity between the items, and (ii)

combining them to compute the similarity between a basket of items and a candidate recommendation. However, these approaches create a number of complications when learning to review texts with repeated parts.

C. Sequential and Temporal Models

Wang-Cheng Kang et al. (2018) proposed a self-attention based sequential model (SASRec), that outperformed various state-of-the-art sequential models (including MC/Convolution Neutral Network (CNN)/ Recurrent Neural Network (RNN) based approaches) on both sparse and dense datasets on data from Amazon, Steam and MovieLens. At each time step, SASRec seeks to identify whether items are relevant from a user's action history. Qiwei Chen et al. (2019) proposed to use the Transformer model to capture the sequential signals underlying user behavior sequences, which performed better than wide and deep learning (WDL) from Google and Deep Interest networks (DIN) from Alibaba on reviews collected from Taobao. Compared to the previous Embedding & Multilayer Perceptron (MLP) paradigm, they added a transformer layer between behavior sequence transformer (BST) and WDL to learn better representations for users' clicked items. Hao Ding et al. (2022) constructed a latent semantic model based on user comments. The adaptive temporal weight of multiple preference features is also improved to calculate the preferences of users at different time periods using human forgetting features, item interest overlap, and similarity at the semantic level of the review text to improve the accuracy of sparse evaluation data. The results show that the accuracy is better than TimeSVD++, BTMF, and other recent models on nine datasets from Amazon.

D. Specialized Learning and Optimization Models

Xiangnan He et al. (2020) proposed a model named LightGCN, including neighborhood aggregation in graph convolution network (GCN) for collaborative filtering, and is much easier to implement and train, exhibiting substantial improvements over Neural Graph Collaborative Filtering (NGCF) under the same experimental setting. Steffen Rendle et al. (2010) proposed a factorized personalized Markov chains (FPMC) model and outperformed the factorized Markov chains method and Markov chains (MC) on both sparse and dense online shopping dataset. The approach factorizes the transition cube with a pairwise interaction model and adapts the Bayesian Personalized Ranking (BPR) framework for sequential basket data to learn parameters. Yong-Guang Chen et al. (2022) proposed Intent Contrastive Learning (ICL), improved robustness against data sparsity and noisy interaction issues, and employed to four data sets from Amazon and Yelp. The core idea is to learn users' intent distribution functions from unlabeled user behavior sequences via clustering and optimize Sequence Recommendation (SR) models with contrastive self-supervised learning (SSL). Keqin Bao et al. (2023) proposed a tuning framework to align the Large Language Model (LLM) with the recommendation framework (TALLRec), significantly enhanced the capabilities of LLM on BookCrossing and MovieLens dataset. TALLRec comprises alpaca tuning, a common training process of LLM that enhances generalization ability, and recommendation tuning which emulates the pattern of instruction tuning and tunes LLMs for the recommendation task.

III. Data Exploration

This project's data exploration phase aimed to refine and extract relevant information for the predictive task of recommending highly rated businesses within a specific geographic region. The Google Local Data (2021) California Ratings only dataset comprised Google Maps reviews within California up to September 2021,

featuring user ratings, timestamps, business metadata, and geographic information. Given the project's focus on recommending restaurants based on user preferences and proximity, it was essential to construct a dataset that accurately represented businesses within a manageable geographic area, ensuring high relevance to real-world applications.

Initially, businesses were filtered using their postal codes to identify those located in San Diego County. However, this approach encountered limitations, as some businesses lacked postal code or address information. To solve this, the filtering process was turn to rely on geographic latitude and longitude. A radius of 10 miles around San Diego's city center was initially chosen, and calculated using the following Haversine formula to determine distances, where we input R as the radius of the Earth.

$$a = \sin^2\left(\frac{\Delta \text{lat}}{2}\right) + \cos(\text{lat}_1) \cdot \cos(\text{lat}_2) \cdot \sin^2\left(\frac{\Delta \text{lon}}{2}\right) \tag{1}$$

$$c = 2 \cdot \arctan 2\left(\sqrt{a}, \sqrt{1-a}\right) \tag{2}$$

$$d = R \cdot c \tag{3}$$

While this provided a more comprehensive set of businesses, the volume was too large for efficient processing. To further streamline the dataset, the radius was reduced to 5 miles from the San Diego city center. Using the refined Haversine formula, metadata for businesses within this smaller range was extracted and cleaned to remove duplicate entries, leaving only unique businesses located within the 5-mile radius. This precise filtering ensured that the metadata represented a targeted set of non-repeated restaurants, aligning closely with the project's objectives.

The final step contains integrating the filtered metadata with the review data. By iterating through the whole California review dataset, reviews corresponding to the gamp_id of businesses in the San Diego 5-mile meta dataset were identified, extracted, and saved into a more focused review dataset, containing all user reviews associated with businesses in the defined range, which accounts for approximately 2.5% of the original California review only dataset. This cleaned and targeted dataset serves as the foundation for the predictive task of the model, capturing businesses and reviews that are most aligned with user preferences within a practical geographic scope.

IV. Task Description

This project focuses on a predictive task to recommend highly-rated restaurants to users based on their preferences, temporal behaviors, and geographic info. The dataset comprises Google Maps reviews in California up to September 2021, including user ratings, timestamps, and geographic data. Thus, combined with real-life situations, we find that it is meaningful to study this predictive task that gives user-recommended restaurants assuming this user was trying to find good restaurants around him in Sep 2021. The ultimate goal is to provide a list of highly recommended restaurants to the user based on taste and personal preference within a certain distance. The major challenge is to accurately predict user ratings for unseen restaurants, leveraging temporal trends, user preferences, and item popularity. The main procedure of our method can be listed in Fig 1.

A. Data Processing and Feature Engineering

To ensure the dataset aligns with the project's objectives, the following preprocessing steps were applied.

1) **Data Sampling:** Businesses were filtered based on geographic proximity to a central location (e.g., San Diego).
2) **Filtering and Cleaning:** Duplicate reviews and noisy data (e.g., missing ratings or invalid timestamps) were removed.

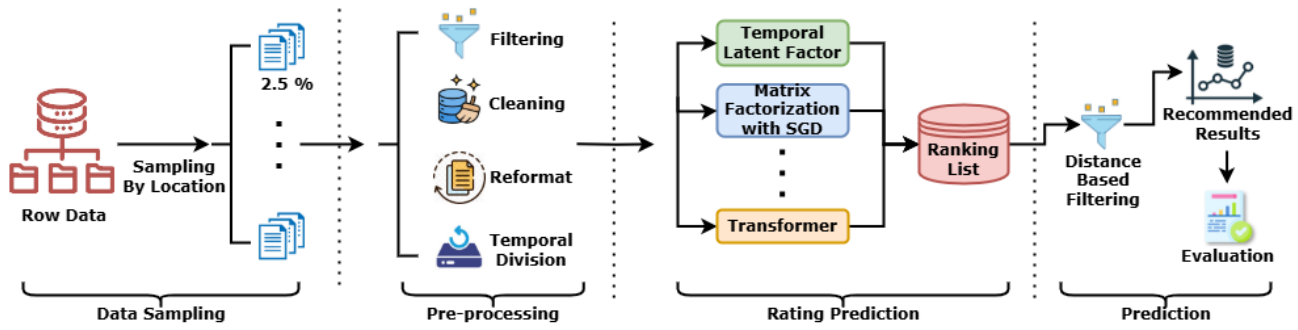

Fig. 1: flowchart

3) **Reformatting:** User and item identifiers were mapped to unique indices, and timestamps were normalized into meaningful units (e.g., days).
4) **Temporal Division:** The dataset was split into training, validation, and test sets, reserving September 2021 for evaluation to mimic a real-world temporal recommendation setting.

B. Evaluation Metrics

The evaluation of model performance focuses on both predictive accuracy and the following metrics are used to assess the accuracy of predicted ratings:

- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)

C. Baseline and Advanced Models

To benchmark the predictive task, the following models are implemented:

1) Baseline: Collaborative Filtering (Bias-Only) Model
2) Matrix Factorization - Stochastic Gradient Descent
3) Latent Factor Model
4) Temporal Latent Factor Model
5) Behavior Sequence Transformer (BST)

V. Model

A. Collaborative Filtering (Bias-Only) Model [Baseline]

The baseline model is designed to predict user ratings for businesses, which captures the inherent biases in the data without directly using features like metadata in the prediction. It operates on the principle of isolating the primary factors influencing ratings.

- Main Components:
  - Global Average: Denoted as α, represents the mean of all the ratings in the training dataset. The formula is shown as follows:

$$\alpha = \frac{\sum_{(u,b) \in training} r_{(u,b)}}{N}$$

    where N denotes the total number of ratings in training set and $r_{(u,b)}$ denotes the rating of user u to business b in training set.
  - User Bias: Denoted as β_u, represents the deviation of ratings given by a random user from the global average α. The formula is shown as follows:

$$\beta_u = \frac{\sum_{b \in A} (r_{u,b} - (\alpha + \beta_b))}{\lambda + \#u}$$

    where A denotes the set of Businesses rated by user u, $\#u$ denotes the number of ratings from user u and λ denotes the regularization term to prevent overfitting.
  - Business Bias: Denoted as β_b, represents the deviation of ratings of a random business from the global average α. The formula is shown as follows:

$$\beta_b = \frac{\sum_{u \in B} (r_{u,b} - (\alpha + \beta_u))}{\lambda + \#b}$$

where B denotes the set of Businesses rated by user u, $\#b$ denotes the number of ratings for business b and λ denotes the regularization term to prevent overfitting.

- Optimization: The model is optimized by alternating updates for α, β_u, and β_b over multiple iterations until convergence, based on the previous three equations.
- Prediction & Evaluation: After optimized, the predictions for user u and business b are calculated as follow:

$$Prediction_{(u,b)} = \alpha + \beta_u + \beta_b$$

The model's accuracy can be evaluated using Mean Squared Error(MSE):

$$\text{MSE} = \frac{\sum_{(u,b)\in\text{Validation}} \left(r_{u,b} - \text{Prediction}_{(u,b)}\right)^2}{\#\text{ Validation Ratings}}$$

- Strengths and Weaknesses
  - Strength:
    - Simple and easy to implement
    - Use regularization parameters to prevent overfitting.
  - Weakness:
    - Ignores metadata and interaction features.
    - Cannot be utilized to model complex relationships.

B. Matrix Factorization - Stochastic Gradient Descent

This model implements a matrix factorization(MF) approach for collaborative filtering using Stochastic Gradient Descent(SGD). It decomposes the user-business interaction matrix into latent factors that predict unseen relationships, also with biases for users and businesses.

- Matrix Factorization: Predicts missing values in a user-item interaction matrix. Decomposes the matrix into two smaller matrices to represent latent factors for users and items in order to predict their implicit features.
  - Interaction Matrix: The rows represent users and columns represent items, each entry represents the interaction. Note that most entries are missing due to sparsity.
  - Latent factors:
    - User latent factors: Denote as p_u, which predicts the preference of a user.
    - Business latent factors: Denote as q_b, which encodes the feature of a business.
- Stochastic Gradient Descent: An optimization algorithm that minimizes the loss function. We use it to minimize the prediction error here.
  - Objective/Loss Function:

$$\text{Loss} = \sum_{(u,i)} (r_{u,i} - \hat{r}_{u,i})^2 + \lambda \left(\|p_u\|^2 + \|q_i\|^2\right)$$

where $r_{u,i}$ denotes the true rating, $\hat{r}_{u,i}$ denotes the predicted rating and λ denotes the regularization parameter to prevent overfitting.

  - Gradient Descent:

$$\theta \leftarrow \theta - \eta \cdot \nabla \text{Loss}$$

where θ is the parameter to optimize, η is the learning rate or step size.

- Main Component:
  - Global bias: Denotes as g, which represents the average rating of all users and businesses.
  - User bias: Denotes as b_u, which represents the tendency of a user u, whose rating tends to be higher or lower.
  - Business bias: Denotes as b_b, which represents the tendency of a business b, where the rating received tends to be higher or lower.
  - Latent factors: p_u and q_b discussed before.

- Optimization: Minimize the loss function:

$$\text{Loss} = \sum_{(u,i)} (r_{u,i} - \hat{r}_{u,i})^2 + \lambda \left(\|p_u\|^2 + \|q_i\|^2 + b_u^2 + b_i^2 \right)$$

- Prediction:

$$\hat{r}_{u,i} = g + b_u + b_b + p_u \cdot q_b$$

- Strengths and Weaknesses:
  - Strength:
    - Can predict the explicit latent relationships and biases.
    - Can avoid overfitting due to regularization.
    - Efficient in computing due to SGD.
  - Weakness:
    - May miss contextual information, such as metadata.
    - Requires careful tuning, due to relying on hyper-parameters, such as eta.

C. Latent Factor Model

The Latent Factor Model is a foundational approach in recommendation systems, which aims to predict user-item interactions by mapping users and items into a shared latent feature space.

- **Model Formulation:** The predicted rating for a user u on an item i is expressed as:

$$r_{u,i} = \mu + \beta_u + \beta_i$$

  where:
  - μ: The global average rating.
  - β_u: User-specific bias, representing user u's deviation from the global average.
  - β_i: Item-specific bias, reflecting the deviation of item i's ratings from the global average.
- **Strength:**
  - Simple, reliable, and computationally efficient.
- **Weakness:**
  - Lack of ability to handle dynamic user behaviors or temporal variations.

D. Temporal Latent Factor Model

The Temporal Latent Factor model extends traditional latent factor models by introducing temporal dynamics into the recommendation process. This approach can capture changes in user preferences and item popularity over time, enabling the model to make more contextually accurate predictions in time-series data. Our prediction task involves historical data with timestamp attributes. Therefore, adding a time bin mechanism to handle future trends should have a positive impact on prediction accuracy. Following are the details of the formula:

- **Model Formulation:** The predicted rating for a user u on an item i at time t is given by:

$$b_{u,i}(t) = \alpha + \beta_u + \alpha_u \cdot \text{dev}_u(t) + \beta_{u,t} + (\beta_i + \beta_{i,\text{Bin}(t)}) \cdot c_u(t)$$

  where:
  - α: The global average rating.
  - β_u: Static user bias, representing the user's general deviation from the global average.
  - α_u: User-specific scaling factor, modulating the impact of temporal deviations.
  - $\text{dev}_u(t)$: Time-dependent deviation function for user u.
  - $\beta_{u,t}$: Short-term bias specific to user u at time t.
  - β_i: Static item bias, representing the item's general deviation from the global average.
  - $\beta_{i,\text{Bin}(t)}$: Temporal item bias, accounting for gradual changes over time bins.
  - $c_u(t)$: Time-dependent scaling factor for user u, defined as $c_u + c_{u,t}$ to include both long-term and short-term effects.
- **Latent Factor Dynamics:** To enhance the model's ability to represent user preferences, latent factors are defined to evolve with time. The temporal evolution of the user latent factor $\gamma_{u,k}(t)$ in dimension

k is expressed as:

$$\gamma_{u,k}(t) = \gamma_{u,k} + \alpha_{u,k} \cdot \text{dev}_u(t) + \gamma_{u,k,t}$$

where:

- $\gamma_{u,k}$: Static user latent factor in dimension k.
- $\alpha_{u,k}$: Factor-specific drift coefficient for user u in dimension k.
- $\gamma_{u,k,t}$: Short-term user factor variation in dimension k.

- **Strength:**
  - Captures complex temporal dependencies in user preferences and item popularity, incorporating both long-term trends and short-term variations.
- **Weakness:**
  - Requires sufficient temporal data to estimate fine-grained temporal biases.

E. Behavior Sequence Transformer

This Behavior Sequence Transformer (BST) is a SOTA recommender architecture based on seq2seq models, which translate user behavior into sequences and predict a rating for each target item It employs a Transformer architecture to model sequential dependencies. Unlike the Temporal Latent Factor model, BST focuses on learning representations from sequences of user-item interactions.

- **Sequential Representation:** User behavior sequences are modeled as ordered interactions, to capture temporal dependencies and evolve preferences. Each sequence consists of a fixed-length window of past interactions.
- **Self-Attention Mechanism:** The core component of BST is the multi-head self-attention mechanism, which computes weighted attention scores to determine the relevance of past interactions for the current prediction.
- **Embedding Layer:** Each user, item, and positional index are embedded into a continuous vector space. The embedding incorporates:
- **Prediction Layer:** The output of the Transformer encoder is pooled and concatenated with the target item embedding for final rating prediction:

$$Pred_{(u,b)} = f(\text{Concat}(\text{SeqEmbed}, \text{ItemEmbed}))$$

where f is a feed-forward neural network.

- **Strength:**
  - Captures sequential dependencies in user behavior through self-attention mechanisms.
  - Learns rich representations of user-item interactions by encoding behavioral sequences.
  - Effectively models both global and contextual user preferences, improving recommendation quality.
- **Weakness:**
  - Extremely High computational cost due to the quadratic complexity of self-attention, especially for long sequences.
  - Requires extensive user interaction history to fully utilize sequence-based features.
  - Sensitive to hyperparameter tuning, including sequence length and attention configurations.

VI. Results Analysis

This section presents the experimental evaluation of the proposed models to predict user ratings and generate restaurant recommendations. The evaluation metrics include Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE), providing a comprehensive assessment of prediction accuracy. Table I summarizes the performance of various models.

Including latent factors significantly improves performance over the baseline, as demonstrated by the Latent Factor Model and MF with SGD. The Matrix

Model	MSE	RMSE	MAE
Collaborative Filtering	1.1556	1.0750	0.7420
MF with SGD	1.1325	1.0642	0.7292
Latent Factor	1.1248	1.0605	0.7391
Temporal Latent Factor	1.3339	1.1550	0.8992
BST	1.1059	1.0516	0.7237

TABLE I: Evaluation Metrics for Different Models

Factorization with the SGD model performs better than the baseline model in terms of all metrics. It has lower MSE indicating a better fit to the data. Similarly, lower RMSE and MAE also represent more accurate and consistent predictions. This is because of the ability to capture latent interactions. The latent factor model further refines this approach, achieving a lower MSE and RMSE, although the MAE is slightly higher. The BST model achieves the best overall performance, indicating its ability to capture more complex patterns in data.

The temporal latent factor model does not enhance accuracy. This may be because the data lacks adequate temporal signals, leading to overfitting to noise or even failure to generalize. An alternative explanation is sparse temporal data poses difficulty for the model to learn temporal trends for users and businesses.

Sequence-based modeling with BST outperforms all other approaches, highlighting its ability to capture user-item interactions and temporal dependencies. The model benefits from its self-attention mechanism, which effectively encodes sequential interactions.

VII. Conclusion

In this work, we conducted a comprehensive comparison and systematic evaluation of five recommendation system methods on Google Local Data (2021), demonstrating the effectiveness of advanced recommendation models for temporal and sequence-based tasks. While simple approaches like collaborative filtering provide reasonable baselines, incorporating latent factors and sequential modeling offers substantial improvements. The BST model, in particular, achieves state-of-the-art performance, making it well-suited for real-world recommendation systems where user preferences evolve over time. However, the features used in our experiments were restricted to user ratings and timestamps, which may not fully capture the complexities of user preferences. Future work could explore hybrid models combining the strengths of LFM and BST for enhanced temporal modeling and integration of techniques to analyze text information.

References

[1] Sarwar, B., Karypis, G., Konstan, J., Riedl J.. 2001. Item-based collaborative filtering recommendation algorithms. Proceedings of the 10th international conference on World Wide Web, pp. 285-295.

[2] Koren, Y., Bell, R., Volinsky, C.. 2009. Matrix Factorization Techniques for Recommender Systems. Computer, vol. 42(8), pp. 30-37.

[3] Michael, J. Pazzani, Daniel Billsus. 2007. Content-Based Recommendation Systems. The Adaptive Web, Springer, pp. 325-341.

[4] Wang-Cheng Kang, Julian McAuley. 2018. Self-Attentive Sequential Recommendation, 2018 IEEE International Conference on Data Mining (ICDM).

[5] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, Wenwu Ou. 2018. Behavior Sequence Transformer for E-commerce Recommendation in Alibaba, Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data Article, No. 12, pp. 1-4.

[6] Hao Ding, Qing Liu, Guangwei Hu. 2022. TDTMF: A recommendation model based on user temporal interest drift and latent review topic evolution with regularization factor. Information Processing & Management, vol. 59(5), pp. 285-295.

[7] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 639-648

[8] Steffen Rendle, Christoph Freudenthaler, Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket

recommendation. Proceedings of the 19th international conference on World wide web, pp. 811-820.

[9] Yongjun Chen, Zhiwei Liu, Jia Li, Julian McAuley, Caiming Xiong. 2022. Intent Contrastive Learning for Sequential Recommendation, Proceedings of the ACM Web Conference 2022, pp. 2172-2182.

[10] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, Xiangnan He. 2018. TALLRec: An Effective and Efficient Tuning Framework to Align Large Language Model with Recommendation, Proceedings of the 17th ACM Conference on Recommender Systems, pp. 1007-1014.