National Research University Higher School of Economics

Faculty of Computer Science

Programme: Data Science and Business Analytics

BACHELOR'S THESIS

Research Project

Product Recommendation Systems in E-commerce using Machine and Deep Learning Algorithms

Prepared by the student of Group 213, Year 4,

Shutov Alexey Andreevich

Thesis Supervisor:

Senior Lecturer, School of Applied Mathematics,

Titova Natalia Nikolaevna

Moscow

2025

Abstract.....3
Аннотация.....4
List of key words.....5
1 Introduction.....5
1.1 Background.....5
1.2 Problem Statement.....6
1.3 Objectives.....6
2 Literature Review.....6
3 Datasets.....9
4 Metrics.....11
5 The choice of the solution methods.....13
5.1 Recommender Net with Label Encoder (сделано).....13
5.2 Content-Based Recommender System using BM25 and Ratings (сделано).....15
5.3 Content-Based Filtering with TF-IDF (сделано).....17
5.4 SASRec and SVD Hybrid Recommender(сделано).....19
5.5 NCF using Microsoft Recommenders Library(сделано).....20
5.6 LightGBM [сделано].....22
5.7 Content-Based Book Recommender Using BERT Embeddings(Сделано) 24
5.8 CatBoost for Google Maps dataset(сделано).....25
Results.....27
Conclusion.....30
References.....32
Appendices.....34

Abstract

This work explores, utilizing machine learning and deep learning approaches, the design, implementation, and assessment of several recommender system models. Understanding the benefits and constraints of various recommendation algorithms is crucial as specific distribution of content expands ever more vital on e-commerce and digital platforms. Applied across several datasets such MovieLens, Goodreads, and Google Maps reviews, the paper investigates several methods including content-based filtering, collaborative filtering, neural networks, and gradient boosting.

To evaluate the models' performance in top-N recommendation situations, ranking performance measures take front stage. The best recommendation quality is shown by hybrid models including sentiment analysis and contextualized features—especially a LightGBM-based model improved with VADER sentiment scores. Strong performance is shown by neural collaborative filtering as well as by conventional models such as BM25 when supported by structured data.

The study offers useful understanding of when particular algorithms are most useful, how data features affect model results, and what improvements might increase actual applicability. These results support a more informed development, assessment, and choice of recommender systems for various industries.

Аннотация

Это исследование посвящено разработке, реализации и оценке различных моделей рекомендательных систем с использованием методов машинного обучения и глубинного обучения. По мере того как персонализированная подача контента становится всё более важной для электронной коммерции и цифровых платформ, понимание сильных и слабых сторон различных алгоритмов рекомендаций приобретает особую актуальность. В рамках работы рассматриваются несколько моделей — включая контентную фильтрацию, коллаборативную фильтрацию, нейронные сети и градиентный бустинг — применяемых к разнообразным наборам данных, таким как MovieLens, Goodreads и отзывы Google Maps.

Особое внимание уделяется метрикам ранжирования для оценки эффективности моделей в сценариях top-N рекомендаций. Результаты показывают, что гибридные модели, сочетающие анализ тональности и контекстные признаки — особенно модель на основе LightGBM, дополненная тональностными оценками VADER — обеспечивают наивысшее качество рекомендаций. Нейронная коллаборативная фильтрация также демонстрирует высокую эффективность, в то время как традиционные модели, такие как BM25, показывают конкурентные результаты при наличии структурированных метаданных.

Исследование предоставляет практические рекомендации о том, когда те или иные алгоритмы наиболее эффективны, как характеристики данных влияют на результаты моделей и какие улучшения способствуют их применимости в реальных условиях. Эти выводы способствуют более обоснованному подходу к построению, оценке и выбору рекомендательных систем для различных областей.

List of key words

- Recommender systems
- LightGBM
- NCF
- SASRec
- Content-Based filtering
- User-Based Filtering
- Catboost
- Sentiment analysis
- Product recommendations

1 Introduction

1.1 Background

Online e-commerce platforms have boomed during the last 5 years. Everyone is watching Netflix, buying products from Amazon or Wildberries, watching videos on YouTube. All these have become a result of the personalized supply system of products to the customers. People get what they are interested in. Now it is common to call these systems as Recommender algorithms. They can take into various types of data depending on the result or aim. For example, 80% of movies on Netflix are watched as a result of work of the recommender systems. Such systems play a significant role in overall sales of the e-commerce platforms.

Despite the fact that recommender systems have proved to be an effective instrument to boost the sales. They have some key factors which might limit their performance and effectiveness.

1.2 Problem Statement

Various recommender algorithms can have their limitations. The goal of the study is to conduct the research when a specific model should be used and what factors may affect its performance. In addition to that suggest improvements to be made. Give an insight into what data is the most appropriate for a chosen model.

1.3 Objectives

- Learn the fundamentals of the recommender systems
- Practice building the recommender models ML/DL
- Study the way of models' evaluation
- Get the intermediate results
- Compare the results of the students' work with the related researches.
- Study the ways to enhance the recommender models

2 Literature Review

Recommender systems are tools used to suggest items, products, or services to users based on various factors such as their preferences, past behavior, or characteristics of the items themselves. These systems are highly relevant in today's digital world, powering platforms like e-commerce websites, music streaming services, and social media networks. There are three primary types of recommendation approaches: Collaborative Filtering, Content-Based Filtering, and Hybrid Methods, as proposed by Adomavicius and Tuzhilin [1].

Collaborative filtering is a popular method that leverages the collective preferences of users to make recommendations. It comes in two forms: memory-based and model-based. Memory-based collaborative filtering involves

computing similarities between users or items and suggesting items based on these similarities. In user-based collaborative filtering, recommendations are made by finding similar users and suggesting items they liked, as described by Sarwar et al. [2]. In item-based collaborative filtering, recommendations are based on items similar to those the user has interacted with, which is another approach highlighted by Sarwar et al. [2]. Model-based collaborative filtering, on the other hand, includes techniques like matrix factorization, where the system decomposes user-item interactions into lower-dimensional matrices, capturing hidden patterns between users and items, a method proposed by Koren et al. [3].

Content-based filtering makes recommendations based on the characteristics of the items and the preferences of the users. In this approach, items are described using features, such as genre, author, or keywords, and recommendations are made by finding items similar to those the user has interacted with in the past. Lops, De Gemmis, and Semeraro discussed this in the context of content-based filtering methods for recommending items based on their inherent attributes [4]. Content-based methods help solve the cold start problem for new users but are limited in that they tend to recommend similar items, reducing diversity in suggestions, as noted by Lops et al. [4].

Hybrid methods combine collaborative filtering and content-based filtering techniques to overcome the shortcomings of each individual approach. A weighted hybrid method would combine the scores from both models, while a switching hybrid approach might switch between models depending on the context or the availability of data. For example, Netflix uses hybrid methods, incorporating both collaborative and content-based strategies to provide diverse recommendations, as described by Gomez-Uribe and Hunt [5].

Machine learning approaches to recommender systems involve using algorithms that can learn patterns from data. Supervised learning models, such as classification and regression techniques, are used to predict user preferences based on labeled data. These models learn from historical user behavior to predict ratings or the likelihood of interaction with an item. Unsupervised learning, including clustering techniques like k-means, groups users or items with similar characteristics, enabling recommendations based on these groupings, as described by Agarwal et al. [6]. More advanced models include reinforcement learning, which optimizes recommendations by learning from user interactions in a dynamic, reward-based environment, as proposed by Liang et al. [7].

Deep learning approaches in recommender systems have gained popularity in recent years due to their ability to model complex patterns. Deep neural networks, such as autoencoders, are used for collaborative filtering, where they learn compressed representations of users and items. Neural collaborative filtering (NCF) is an example of a model where a neural network learns the interaction between users and items, enhancing the predictive accuracy of the recommendations, as proposed by He et al. [8]. Recurrent neural networks (RNNs) are used for sequential recommendations, as they capture temporal patterns in user behavior, such as recommending content based on past interactions over time, as demonstrated by Hochreiter and Schmidhuber [9].

Evaluating recommender systems involves measuring how well the system's predictions align with actual user behavior. Common accuracy-based metrics include root mean square error (RMSE) and mean absolute error (MAE), which measure the difference between predicted and actual ratings. Ranking-based metrics such as precision, recall, and F1-score are used to evaluate top-N recommendations, where a system suggests a list of items to the user. These metrics are widely discussed by Serrano et al. [10]. Additional metrics like hit

rate, coverage, and diversity assess how representative or diverse the recommendations are, while business impact metrics focus on conversion rates and user engagement.

Despite their success, recommender systems face several challenges. The cold start problem arises when there is insufficient data for new users or items, making it hard to generate accurate recommendations. Scalability is another issue, as the number of users and items grows, making computations increasingly complex. Additionally, recommender systems often suffer from a lack of diversity, as they tend to recommend similar items over time, which can reduce the novelty of suggestions. Bias and fairness in recommendations is also a growing concern, as algorithms can unintentionally reinforce existing biases, favoring certain users or groups over others, as discussed by Ekstrand et al. [11].

In conclusion, the field of recommender systems has evolved significantly, from traditional collaborative filtering to more sophisticated machine learning and deep learning approaches. Each method has its strengths and weaknesses, and hybrid approaches aim to balance the trade-offs. While challenges remain, the future of recommender systems lies in refining existing models, making them more adaptive, fair, and explainable, while integrating new data sources and algorithms for more personalized and effective recommendations, as envisioned by Zhao et al. [12].

3 Datasets

To develop and evaluate the recommender system, a diverse set of datasets from different domains was utilized, including movies, local business reviews, and

books. This variety allowed for comprehensive testing of the system's performance across varying types of user-item interactions and content structures.

The MovieLens datasets, a standard benchmark in recommender systems research, were employed in two versions: MovieLens 1M and MovieLens 100K. The MovieLens 1M dataset contains approximately one million user ratings and includes four key attributes: user ID, item ID, rating, and timestamp. Similarly, the MovieLens 100K dataset provides 100,000 ratings with additional metadata. In addition to user ID, item ID, rating, and timestamp, it includes the title and genres associated with each movie, as well as user-generated tags that serve as short comments. These datasets provide both explicit user feedback and auxiliary content information, making them well-suited for collaborative and content-based recommendation techniques.

In addition to movie-related data, business review data from Google Maps was used, specifically reviews of business locations in South Carolina. Two variations of this dataset were included. The first is a full review dataset containing user ID, the unique identifier for the business location (gmap_id), star rating, review text, and the time of the review. The second version is a simplified dataset which consists only of user ID, gmap_id, rating, and timestamp, focusing only on structured interactions without textual content. This data supports the modeling of user preferences in a location-based recommendation scenario.

Lastly, a dataset from Goodreads was incorporated to study the book recommendation use cases. This dataset consists of user ID, book ID, an indicator for whether the book was read(action), the rating provided by the user, incomplete review text (serving as a form of user comment), and the date the user started reading the book. This dataset provides a rich basis for testing

hybrid recommendation models by combining elements of explicit feedback with behavioral metadata and user comments.

Together, these datasets provide a multifaceted basis for training, validating, and benchmarking the proposed recommender system across various domains and data structures.

4 Metrics

To assess the performance of the recommendation models, a combination of regression-based and ranking-based metrics was used. For models predicting explicit ratings: Root Mean Squared Error (RMSE) and Mean Squared Error (MSE) were utilized to evaluate the accuracy of numerical predictions. RMSE provides an interpretable measure of the average deviation between predicted and actual ratings, while MSE emphasizes larger errors due to its squared nature, offering insights into the variance of prediction inaccuracies.

Additionally, Mean Absolute Percentage Error (MAPE) was employed to measure the average absolute error as a percentage of actual values, which is particularly useful when comparing errors across datasets with different rating scales.

For evaluating the quality of top-N recommendations, several ranking-based metrics were considered. Precision@K measures the proportion of relevant items among the top-K recommended items, providing an indication of recommendation accuracy. Recall@K evaluates the proportion of all relevant items that appear in the top-K recommendations, focusing on completeness. Normalized Discounted Cumulative Gain (NDCG@K) accounts for both the relevance and position of items in the recommendation list, rewarding methods

that rank highly relevant items higher. Hit Rate@K (or Hit@K) was also used, measuring the proportion of users for whom at least one relevant item appears in the top-K recommendations. Together, these metrics offer a comprehensive evaluation framework, capturing both the predictive accuracy and the effectiveness of the models in generating relevant and well-ranked recommendations.

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

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

$$\text{NDCG@K} = \frac{1}{Z_K}\sum_{i=1}^{K}\frac{2^{rel_i} - 1}{\log_2(i+1)}$$

- Z_k normalising factor ensuring that the value is between 0 - 1
- rel_i relevance score of the item at position i
- K number of top recommendations considered

$$\text{Precision@K} = \frac{\text{Number of relevant items in top-K recommendations}}{K}$$

$$\text{Recall@K} = \frac{\text{Number of relevant items in top-K recommendations}}{\text{Total number of relevant items}}$$

$$\text{Hit@K} = \frac{\text{Number of users with at least one relevant item in top-K recommendations}}{n_{\text{users}}}$$

5 The choice of the solution methods

Deep learning methods combined with machine learning were conducted.

5.1 Recommender Net with Label Encoder

This part presents a hybrid recommendation approach applied to the MovieLens 100K dataset, combining collaborative filtering with content-based metadata such as movie genres and user-generated tags. A Multi-Layer Perceptron (MLP)-based model was employed to learn user preferences by integrating user-item interactions and content features.

Data Preprocessing

The dataset includes user ratings, movie information (titles, genres, and external identifiers), and free-form tags. Movie metadata was merged using movie IDs across the ratings, movies, links, and tags tables. Tags were aggregated by movie and concatenated into a single string per item. Missing tags were imputed with empty strings. Genres were transformed from pipe-delimited strings into binary multi-hot vectors using the MultiLabelBinarizer, following a common practice for handling multi-label categorical features in recommender systems.

User and item IDs were encoded using LabelEncoder to convert them into integer indices suitable for embedding layers. This approach reduces the sparsity inherent in large ID spaces and facilitates efficient model training. The

resulting dataset included encoded user and item IDs, genre encodings, and the target rating variable.

The dataset was split into training and validation sets using an 90/10 stratified split. This was done to maintain rating distribution and ensure consistent generalization during model evaluation.

Model Architecture

The proposed model, RecommenderNet, is a neural network built using PyTorch. It utilizes an MLP architecture with the following components:

- User Embedding: Projects each user ID into a 64-dimensional dense vector.
- Item Embedding: Projects each item ID into a 64-dimensional dense vector.
- Genre Input: A multi-hot binary vector representing genres for each item.

These components are concatenated into a single vector and passed through two fully connected layers. The first layer contains 128 hidden units with ReLU activation; the second is a single output neuron predicting the rating score. The architecture is inspired by prior work on neural collaborative filtering, which has demonstrated the effectiveness of deep learning in capturing nonlinear user-item interactions.

Training Procedure

The model was trained for five epochs using the Adam optimizer with a learning rate of 0.001. The loss function used was Mean Squared Error (MSE), commonly adopted for regression-based recommendation tasks. Input data was batched and processed via a custom PyTorch Dataset class, ensuring efficient loading and GPU compatibility. No explicit regularization techniques such as

weight decay or early stopping were used, though dropout could be integrated in future extensions.

Evaluation Metrics

The model was evaluated using both regression and ranking metrics. First, regression performance was assessed on the validation set using Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), which are standard measures for predicting numeric ratings.

Next, ranking performance was measured using Precision@10, Recall@10, Normalized Discounted Cumulative Gain (NDCG@10), and Mean Average Precision (MAP@10). Ground truth was defined as items rated 4 or higher, following established conventions for treating high ratings as positive interactions. For each user, predictions were made over the full item set, and the top-10 predicted items were compared to the user's ground truth set. This full-rank evaluation avoids sampling bias and better reflects recommendation performance in realistic settings.

5.2 Content-Based Recommender System using BM25 and Ratings

This experiment uses content-based recommendation techniques to generate movie recommendations based on metadata (genres and tags) using the BM25 algorithm, enhanced with movie ratings. The MovieLens dataset is utilized, incorporating movies, ratings, and tags to create a recommendation system that utilizes both content-based filtering and rating-based boosting.

The preprocessing steps involved:

1. Merging Tags with Movies: The tags dataset was aggregated by movieId, joining all the tags for a given movie into a single string. These were then merged with the movie's dataset.
2. Combining Metadata: The genres and tags were concatenated into a single string, referred to as metadata, for each movie. The genres were separated by spaces, and missing tags were handled by filling NaN with an empty string.
3. Tokenization: The metadata was tokenized using the word_tokenize function from the Natural Language Toolkit (NLTK). The tokenized metadata serves as the input for the BM25 algorithm, which uses the tokenized text for information retrieval.

BM25

A probabilistic information retrieval technique, the BM25 algorithm, ranks documents—in this case, movies—based on the relevance of the query to the contents of the document. Every movie's BM25 score is calculated using its information; subsequently, the scores are modified by including the average rating of every movie.

- BM25 Scoring: Tokenized metadata drives the computation of a given movie's BM25 score. Relevance of any movie is assessed by matching its metadata with a query—that of a target movie.
- Rating Boost: A rating boost increases the BM25 score of every movie to include user tastes. The rating_weight is a variable that modifies how highly ratings affect the recommendation score. Computed as a fraction of the movie's average rating divided by the maximum rating value—5 in this case—the rating increase is

Recommendation Function

The recommendation function, content_based_recommendations, accepts a movie ID and generates the top-N movie recommendations based on the BM25 score and the average rating boost. The steps are as follows:

1. Compute BM25 scores for the target movie.
2. Enhance the BM25 scores by adding the rating boost for each movie.
3. Sort the movies by the enhanced score and return the top-N recommendations, excluding the target movie itself.

To evaluate the recommendation system, the Normalized Discounted Cumulative Gain (NDCG) at 10, Precision at 10, Hit at 10, Recall at 10 were computed.

5.3 Content-Based Filtering with TF-IDF

This experiment applies a content-based recommendation approach to the MovieLens dataset, utilizing TF-IDF (Term Frequency-Inverse Document Frequency) vectorization to analyze the metadata of movies (genres and tags) and combining this with movie ratings. The system recommends movies based on their textual similarity to a target movie, with the added influence of average user ratings.

The dataset consists of the following:

- Movies dataset: Contains movie information, such as movieId, title, and genres.
- Tags dataset: Contains user-generated tags associated with movies, identified by movieId and tag.

- Ratings dataset: Contains ratings provided by users for movies, identified by userId, movieId, and rating.

The preprocessing steps include:

1. Merging Tags with Movies: The tags for each movie are aggregated by movieId, and the tags for a given movie are combined into a single string. This is merged with the movie's dataset.
2. Combining Metadata: The genres and tags for each movie are concatenated into a single metadata string. Genres are separated by spaces, and missing tags are handled by filling NaN with an empty string.
3. TF-IDF Vectorization: TF-IDF vectorizing turns every movie's metadata into a numerical form. TF-IDF indicators, with relation to the body of all movies, the significance of words in the metadata of every movie.

The movie metadata is TF-IDF vectorized to provide a sparse matrix whereby every movie's metadata is expressed as a vector of words. Cosine similarity, a typical metric to assess the resemblance between two non-zero vectors, then compares these vectors. From 0 to 1, the cosine similarity score goes; 1 denotes perfect similarity and 0 denotes no similarity.

Recommendations are generated for a given movie by:

1. Cosine Similarity: Computed between the metadata vector of a target movie (identified by movieId) and that of every other movie in the collection.
2. Rating Boost: To personalize recommendations based on user preferences, the similarity scores are adjusted by incorporating the movie's average rating. A weight factor (rating_weight) is used to scale the influence of the rating.

The rating is normalized to the range [0, 1] by dividing the average rating by 5 (the maximum rating).

3. Top-N Recommendations: The movies are sorted by their enhanced similarity score, and the top-N most similar movies (excluding the target movie itself) are selected as recommendations.

5.4 SASRec and SVD Hybrid Recommender

In this experiment, a matrix factorization approach is applied to the MovieLens dataset using Singular Value Decomposition (SVD).The dataset consists of the information about movies, including movieId and title. The goal is to predict missing ratings for movies by decomposing the user-item interaction matrix into latent factors. The recommendations are then evaluated using standard recommendation metrics such as Hit@K, NDCG@K, Precision@K, and Recall@K.

The preprocessing steps involved:

1) Rating Normalization: The ratings in the dataset are normalized by dividing by the maximum rating (5 in this case). This converts the ratings into a 0-1 scale.
2) User-Item Matrix: A user-item interaction matrix is built using pandas' pivot function, where rows represent users, columns represent movies, and values represent ratings. Missing values (unrated movies) are filled with zeros.

Singular Value Decomposition (SVD)

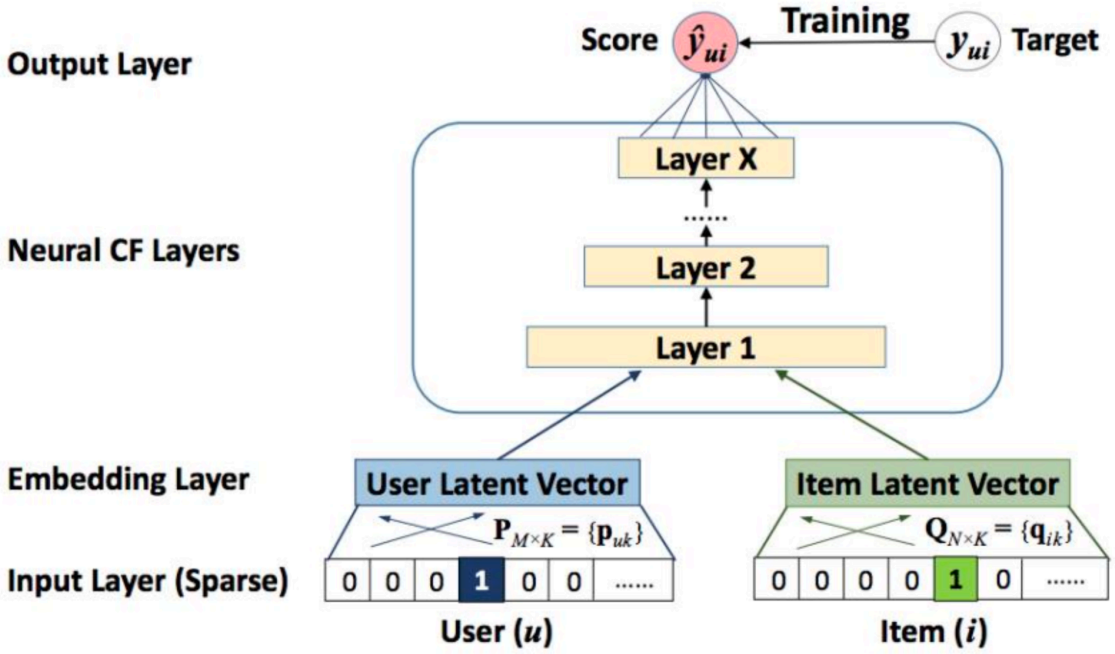
SVD is a matrix factorization technique used to decompose the user-item interaction matrix into three matrices: user factors, item factors, and a diagonal matrix of singular values. In this case, Truncated SVD is used, which reduces

the dimensionality by keeping only the top-k singular values, thereby retaining the most significant latent factors.

Recommendation Generation

The ratings for a target user (user 1 in this case) are extracted from the user-item matrix. The scores for each movie are predicted by computing the dot product between the user's latent factors and the item factors. Movies that the user has already rated are excluded from the recommendations. The top-N recommendations are selected by sorting the predicted scores in descending order and filtering out the movies already rated by the user.

5.5 NCF using Microsoft Recommenders Library



NCF Scheme

This study applies the Neural Collaborative Filtering (NCF) model for recommender systems, using the MovieLens 1M dataset. The model was built on the basis of the Microsoft open source library called recommenders.
The goal of the experiment is to assess the effectiveness of the NCF model, specifically the NeuMF (Neural Matrix Factorization) variant, in predicting user-item interactions and generating top-K recommendations.

The NCF model used in this study is based on the "NeuMF" architecture, a hybrid of matrix factorization and multi-layer perceptron (MLP) (He et al., 2017).

The performance of the model was evaluated using four common ranking-based metrics: MAP@K, NDCG@K, Precision@K, and Recall@K. These metrics were computed for K=10 to assess the top-K recommendation quality.

5.6 LightGBM

Using LightGBM, this section offers a consistent pipeline for rating prediction and tailored recommendations combining transformer-based semantic embeddings with lexicon-based sentiment features. To improve model performance in ranking tasks as well as regression/classification, the method aggregates structured information, time-derived features, and textual analysis.

LightGBM for ranking, classification, and regression

Across several phases—numeric rating prediction, star rating classification, top-K item recommendation—LightGBM serves as the central model. Designed

for speed, memory economy, and scalability, LightGBM is a tree-based gradient boosting system. Perfect for high-dimensional recommendation jobs, it manages big-scale categorical data, provides GPU acceleration, and contains native multiclass and regression goals.

In this pipeline, LightGBM is trained with both structured features and text-derived vectors, then evaluated using RMSE for regression, accuracy/F1 for classification, and Recall@K, Precision@K, Hit Rate, and NDCG for ranking.

Sentiment Analysis with VADER

In the first component of the pipeline, VADER (Valence Aware Dictionary and sEntiment Reasoner) is used to extract sentiment polarity from user-generated review text. VADER is a lexicon- and rule-based tool designed for social media content. It provides compound sentiment scores without requiring labeled data, allowing fast and interpretable sentiment integration into model inputs. These sentiment scores serve as lightweight affective signals during regression and ranking tasks.

Semantic Embeddings with Sentence-BERT

For deeper semantic understanding, the second component of the pipeline leverages Sentence-BERT, specifically the all-MiniLM-L6-v2 model, to encode reviews into dense vector embeddings. Sentence-BERT produces fixed-size sentence-level embeddings that capture the contextual meaning of reviews. These embeddings are precomputed in batches and stored efficiently using NumPy memory maps to support GPU-based model training without exceeding memory limits. Combined with structured features, they enhance the classifier's ability to learn nuanced representations of user opinion and product relevance.

Text Preprocessing and Feature Engineering

Textual data undergoes preprocessing to remove noise such as URLs, user mentions, and punctuation. Features like text length and the number of exclamation marks are extracted as proxies for verbosity and emphasis. In parallel, rich temporal features are engineered from timestamps, including hour of day, weekday, month, time-of-day segment (morning/afternoon/evening/night), weekend indicators, and the number of days since a review was posted. These features help the model capture behavioral and recency-based signals in time-sensitive applications.

Candidate Generation and Ranking Evaluation

To evaluate recommendation performance, the pipeline constructs candidate sets by pairing a positive item with 99 unobserved negatives for each test user, following standard practice in implicit-feedback recommendation evaluation. For each candidate set, LightGBM predicts relevance scores using both structured metadata and embeddings. Metrics such as Recall@K, Precision@K, Hit Rate, and NDCG assess how effectively the model ranks relevant items. This evaluation simulates realistic top-K recommendation scenarios and demonstrates the model's utility in content-aware recommendation systems.

5.7 Content-Based Book Recommender Using BERT Embeddings

This section outlines a content-based recommendation system built using user review data from a Goodreads dataset of comics and graphic novels. The method utilizes semantic embeddings generated from review texts to model user preferences and generate personalized book recommendations.

Data Loading and Preprocessing

The dataset, stored in JSON Lines format, contains fields such as user_id, book_id, and review_text_incomplete. After loading the data, reviews consisting solely of whitespace or empty strings were removed. For each book, all reviews were aggregated by concatenating them into a single text block to represent the overall content associated with the book.

Embedding Generation

To capture the semantic meaning of book reviews, the all-mpnet-base-v2 model from the SentenceTransformers library was employed. This model transforms aggregated textual reviews into dense vector embeddings that preserve contextual relationships and meaning. Each embedding corresponds to a book_id and serves as a fixed-size representation of that book's content.

User Profile Construction and Recommendation

For each user, a profile was generated by computing the element-wise median of the embeddings for books the user had previously reviewed, excluding the test book. This technique assumes that aggregating embeddings can effectively summarize user preferences in latent semantic space. The user profile vector was then compared to all other book embeddings using cosine similarity. Previously interacted books were excluded from the candidate set, and the top-k most similar books were selected as recommendations.

Evaluation Process

A leave-one-out evaluation strategy was employed to assess the recommender system's performance. For each user with at least two interactions, the most recent book was withheld for testing, while the earlier interactions were used to construct the user profile. The system generated a ranked list of recommended books, and evaluation metrics were calculated by comparing the

recommendations to the held-out book. This protocol reflects common practice in recommender systems research for assessing ranking-based performance. The process was repeated across all qualifying users, and the final results were obtained by averaging the evaluation metrics over the user base.

5.8 CatBoost for Google Maps dataset

A JSON file containing user reviews is loaded and converted into a structured DataFrame. Only essential columns—user identifiers, item identifiers (gmap_id), rating scores, and timestamps—are retained. The timestamps are transformed into a normalized time feature to support temporal modeling. The dataset is further enriched through feature engineering. For each user, the number of reviews, the average rating given, and the number of days since their first review are calculated. Similarly, each item is characterized by its number of reviews, average rating received, and the time since its first recorded review. These types of user and item-level statistical features have been widely used in collaborative filtering contexts to enhance model expressiveness and alleviate sparsity issues. Records involving users or items with fewer than three reviews are removed to ensure data quality and stability in model training, a common practice in recommender systems to reduce noise and cold-start effects.

The data is then partitioned into training, validation, and test sets. The training and validation sets are used to build and tune the model, while the test set is reserved for the final evaluation. The model used is CatBoost, a gradient boosting framework that efficiently handles categorical features through techniques such as ordered boosting and target statistics regularization. A CatBoost regressor is trained on both categorical and numerical features to predict the rating a user might give to an item. The model's predictive

performance is initially evaluated using standard regression metrics: root mean squared error (RMSE) and mean absolute error (MAE). These metrics are commonly used in recommender systems to measure the accuracy of predicted ratings.

To assess the system's effectiveness as a recommender, a top-K ranking evaluation is conducted. This evaluation simulates a typical user scenario in which a recommender must suggest a small number of relevant items from a larger pool. For each user in the test set who has at least one highly rated item (defined as a rating of 4 or higher), a single positive item is selected. To simulate a recommendation setting, 99 items that the user has not interacted with are randomly sampled as negative candidates. This approach follows the widely used *leave-one-out* plus negative sampling framework proposed in earlier work on implicit feedback datasets.

For each candidate item, the same user and item features used during training are assembled. The trained model assigns a relevance score to each candidate, and the items are ranked according to these scores. Evaluation metrics such as precision@K, recall@K, normalized discounted cumulative gain (NDCG@K), and hit rate@K are computed. These metrics capture different aspects of recommendation performance, including the ability to rank relevant items highly and the likelihood of including at least one relevant item in the top-K recommendations. Precision and recall offer interpretable measures of recommendation accuracy, while NDCG introduces a position-sensitive view that rewards highly ranked relevant items more than those placed lower in the list.

To offer a strong study of recommender system performance, this method aggregates explicit rating prediction with sampled ranking evaluation. It models complicated user-item interactions and evaluates their practical significance in

ranking situations using contemporary machine learning methods including gradient boosting and feature engineering. Combining predictive accuracy with suggestion quality helps this approach match top standards in comprehensively assessing recommender systems.

Results

Look at the table “Results”. The full amount of obtained metrics one can see in the notebooks of GitHub.

The comparative evaluation of recommendation models across multiple datasets and algorithmic paradigms reveals critical differences in performance, especially when viewed through the lens of ranking-based effectiveness. In practical recommender systems, the ability to identify and prioritize the most relevant items for users—reflected in metrics such as Precision@K, Recall@K, NDCG@K, MAP@K, and Hit Rate—is of central importance. Accordingly, this analysis focuses on top-K recommendation quality as the primary axis of comparison.

Across the evaluated models, the most notable ranking performance was achieved by the LightGBM model augmented with sentiment features derived from user reviews via VADER. This hybrid system attained the highest ranking results, with Recall@10 of 0.2880, Precision@10 of 0.0288, and NDCG@10 of 0.1586. These results confirm that integrating textual sentiment alongside structured metadata significantly enhances a model’s ability to personalize recommendations. Furthermore, the consistent improvement across multiple K-values (e.g., Recall@20 = 0.3132) demonstrates the scalability and robustness of this feature-enriched approach in content-heavy domains like local business reviews.

By contrast, the baseline LightGBM model without sentiment features produced considerably lower metrics (e.g., NDCG@10 = 0.0423, Recall@10 = 0.0845), despite using the same structured inputs. This highlights the importance of affective and semantic signals in distinguishing highly relevant items, particularly in domains where textual reviews offer rich contextual cues.

Among neural approaches, Neural Collaborative Filtering (NCF) using the Microsoft Recommenders framework also yielded strong ranking results. It achieved a Precision@10 of 0.1248, MAP@10 of 0.0558, and NDCG@10 of 0.1354, indicating a solid ability to rank relevant items effectively. While its Recall@10 (0.0728) was below that of the sentiment-augmented LightGBM, the model demonstrated higher precision in surfacing relevant items within the top-most positions of recommendation lists, reflecting its suitability for personalized environments with rich interaction histories.

Traditional content-based filtering methods showed surprisingly competitive performance in ranking metrics, particularly the BM25 model with rating boost, which achieved Recall@10 of 0.1708 and NDCG@10 of 0.1771. This outperformed the TF-IDF-based approach, which nonetheless achieved respectable results (Recall@10 = 0.1544, NDCG@10 = 0.1648). These findings illustrate the continued relevance of information retrieval techniques in recommendation contexts, especially where rich item metadata is available.

In contrast, the deep learning-based RecommenderNet, though producing the best rating prediction accuracy (RMSE = 0.8141, MAE = 0.6165), underperformed in ranking tasks, with Precision@10 of 0.00699 and NDCG@10 of 0.0343. This reflects a common limitation: models trained for point-wise prediction often lack the structural optimization to deliver high-quality ranked outputs.

Similarly, the SVD-based matrix factorization model, enhanced with SASRec-style sequential modeling, showed moderate ranking performance (Recall@10 = 0.15, NDCG@10 = 0.121). While capable of generalization in sparse interaction spaces, its results suggest that latent factor models alone may be insufficient to compete with richer hybrid or deep architectures.

The content-based book recommender using Sentence-BERT embeddings yielded relatively low performance, with NDCG@10 of 0.0299 and Recall@10 of 0.0499. Though semantically rich, the model's absence of behavioral or collaborative inputs perhaps reduced its efficacy. Applied to the same domain as LightGBM, the CatBoost-based method likewise underperformed (NDCG@10 = 0.0166), therefore highlighting the need of deeper contextual modeling outside handmade characteristics.

All told, the study validates that models including behavioral, semantic, and sentiment-driven elements frequently outperform those depending just on structured or latent representations. Although neural and hybrid models such as NCF provide great accuracy and personalization, the combination of LightGBM with sentiment-aware content features produced the best overall recall and ranking performance. These results highlight the need of building recommender systems for top-N recommendation situations not only for predicted accuracy but also for efficient ranking, personalizing, and interpretability.

Conclusion

Recommender systems have become an essential component of today's digital platforms, powering personalized experiences on services like Netflix, Amazon, and YouTube. As this study has shown, such systems significantly impact user satisfaction and business outcomes by delivering relevant content or products. The project set out to explore when specific recommendation models should be applied, what factors influence their performance, and how different types of data affect their outcomes. The goals and objectives of the study were met.

Through the implementation and evaluation of a range of machine learning and deep learning-based recommender algorithms, several key findings emerged. Among all tested models, those that combined structured data with contextual signals—such as the LightGBM model enhanced with sentiment analysis using VADER—consistently delivered the best top-N recommendation performance. This approach achieved the highest ranking metrics, demonstrating that aenriching models with behavioral and emotional cues from user reviews can significantly improve relevance and personalization.

In parallel, Neural Collaborative Filtering (NCF) also performed strongly, especially in precision and mean average precision, suggesting its suitability in scenarios where accurate personalization is crucial and sufficient user-item interaction data is available. BM25 and other conventional content-based models showed good performance when enhanced by rating data, demonstrating that even more basic approaches may work if supplemented by well-organized metadata.

On the other hand, models tuned for rating prediction—such as the deep learning-based RecommenderNet—achieved great regression accuracy but underperformed in ranking tasks, so stressing the need of selecting evaluation metrics fit for the intended use.

This study generally validates that high-quality suggestions depend critically on the type and quality of data as well as on the architecture of the model itself. The studies provide practical knowledge with model creation and evaluation, result interpretation, and identification of doable solutions to raise their performance. These revelations provide a strong basis for next research and a better knowledge of how to match model capabilities with corporate and user objectives in practical recommender systems.

References

1. Adomavicius G., Tuzhilin A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions // IEEE Transactions on Knowledge and Data Engineering. – 2005. – Vol. 17, No. 6. – P. 734–749.
2. Sarwar B., Karypis G., Konstan J., Riedl J. Item-based collaborative filtering recommendation algorithms // Proceedings of the 10th International Conference on World Wide Web. – 2001. – P. 285–295.
3. Koren Y., Bell R., Volinsky C. Matrix factorization techniques for recommender systems // Computer. – 2009. – Vol. 42, No. 8. – P. 30–37.

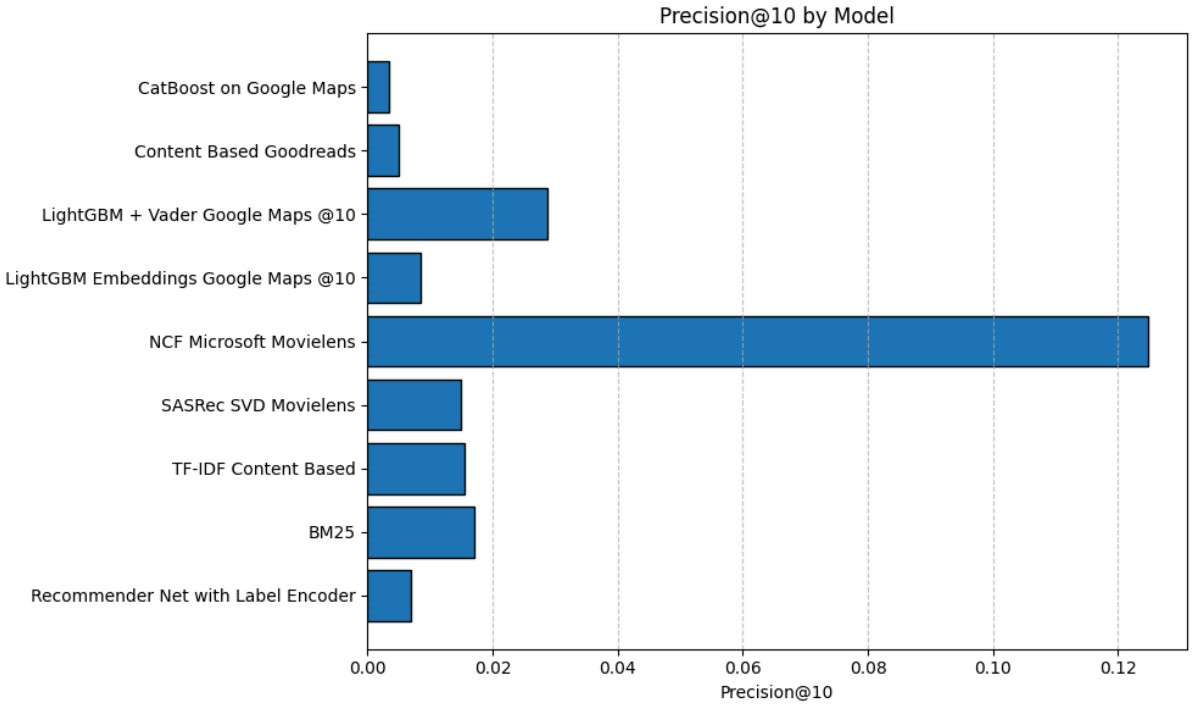
4. Lops P., De Gemmis M., Semeraro G. Content-based recommender systems: State of the art and trends // Recommender Systems Handbook. – Springer, 2011. – P. 73–105.
5. Gomez-Uribe C.A., Hunt N. The Netflix recommender system: Algorithms, business value, and innovation // ACM Transactions on Management Information Systems. – 2016. – Vol. 6, No. 4. – Article 13.
6. Agarwal D., Chen B.C., Elango P. Spatio-temporal models for estimating click-through rate // Proceedings of the 18th International Conference on World Wide Web. – 2009. – P. 21–30.
7. Liang D., Krishnan R.G., Hoffman M.D., Jebara T. Variational autoencoders for collaborative filtering // Proceedings of the 2018 World Wide Web Conference. – 2018. – P. 689–698.
8. He X., Liao L., Zhang H., Nie L., Hu X., Chua T.S. Neural collaborative filtering // Proceedings of the 26th International Conference on World Wide Web. – 2017. – P. 173–182.
9. Hochreiter S., Schmidhuber J. Long short-term memory // Neural Computation. – 1997. – Vol. 9, No. 8. – P. 1735–1780.
10. Serrano E., Iglesias C.A., Garijo M. A survey of metrics for evaluating recommender systems // Artificial Intelligence Review. – 2014. – Vol. 42. – P. 159–192.
11. Ekstrand M.D., Tian M., Azpiazu I.M., Ekstrand J.D., Pera M.S., McNeill D. All the cool kids, how do they fit in?: Popularity and fairness in recommender systems // Proceedings of the 1st Conference on Fairness, Accountability and Transparency. – 2016. – P. 172–186.
12. Zhao X., Zhang W., Wang J., Zhang Y., Rong C., Wen J.R. Deep reinforcement learning for list-wise recommendations // Proceedings of the 28th International Conference on Artificial Intelligence. – 2019. – P. 3590–3596.

Appendices

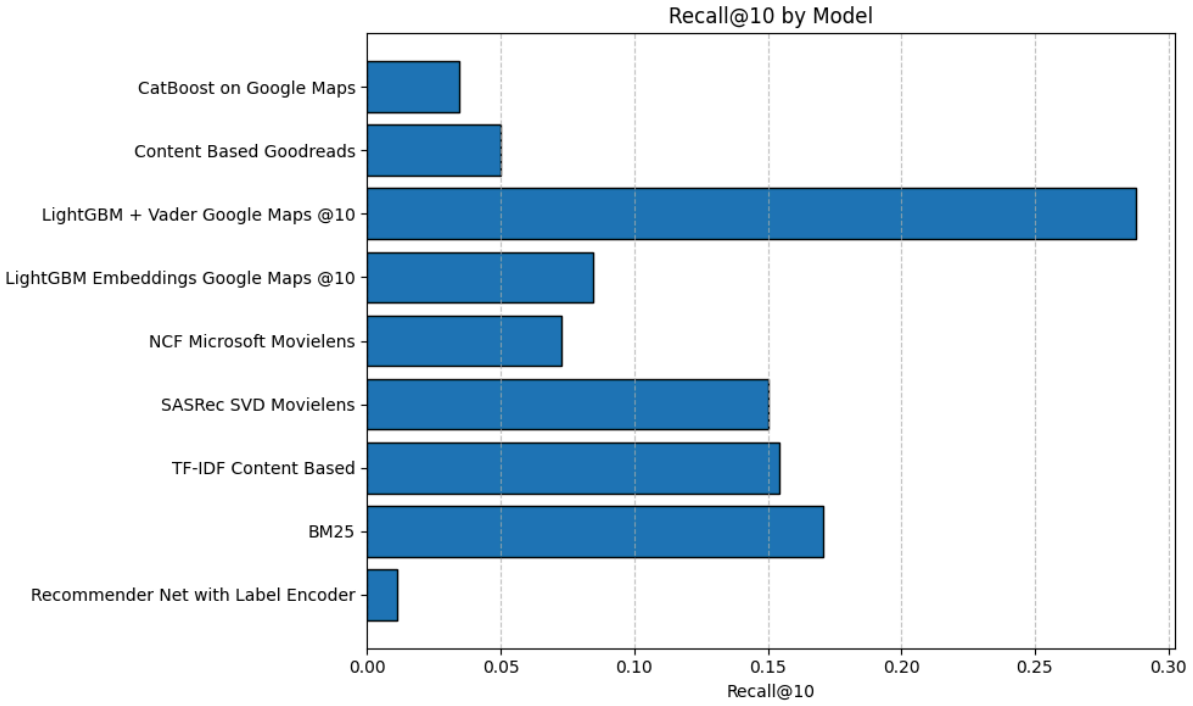
https://github.com/alexey077/diplom

Model	Precision@10	Recall@10	NDCG@10	MAP@10	Hit@10
Recommender Net with Label Encoder	0.007	0.011	0.034	0.005	
BM25	0.017	0.171	0.177		
TF-IDF Content Based	0.015	0.154	0.165	0.073	
SASRec SVD Movielens	0.015	0.15	0.121		0.150
NCF Microsoft Movielens	0.125	0.073	0.135	0.056	
LightGBM Embeddings Google Maps	0.008	0.085	0.042		0.085
LightGBM + Vader Google Maps	0.029	0.288	0.159		0.288
Content Based Goodreads	0.005	0.050	0.030	0.024	
CatBoost on Google Maps	0.003	0.034	0.017		0.034

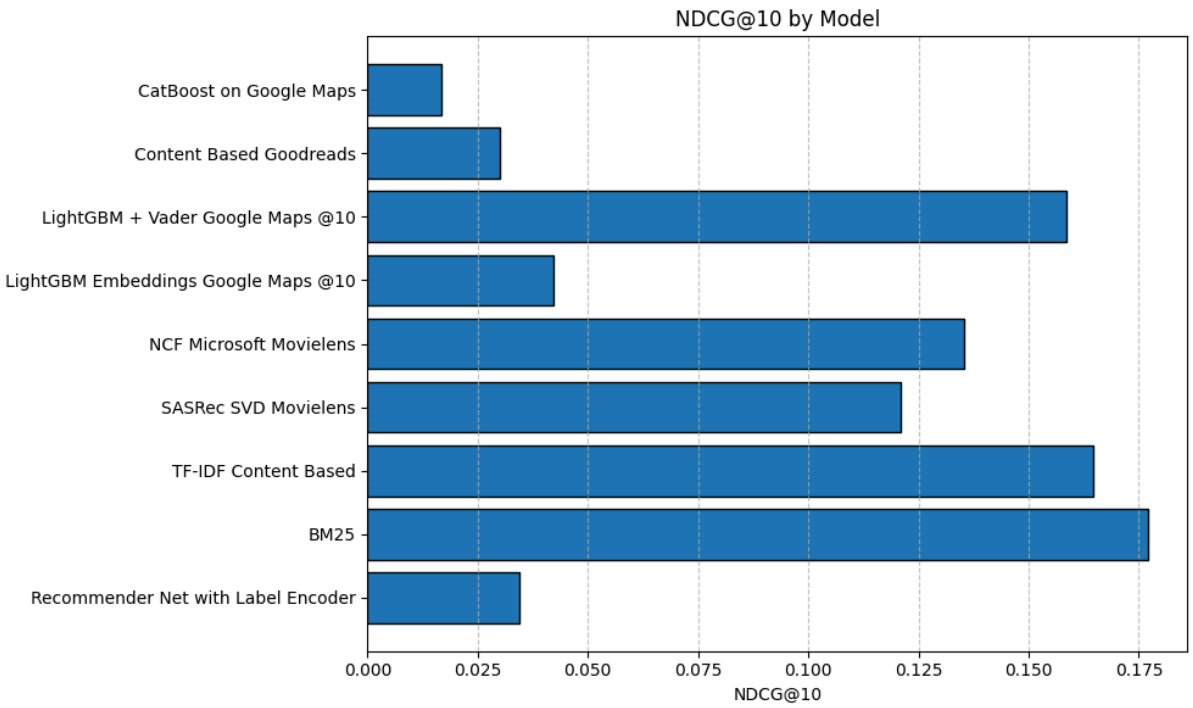
Table “Results”



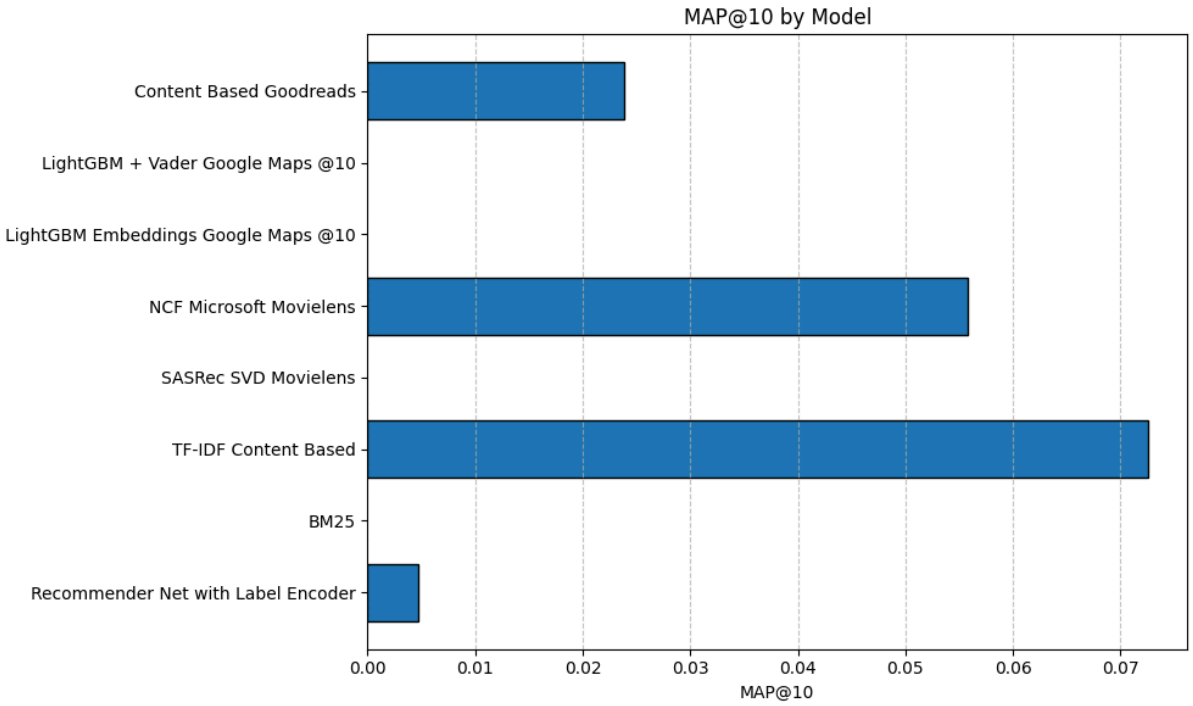
Graph Precision at 10



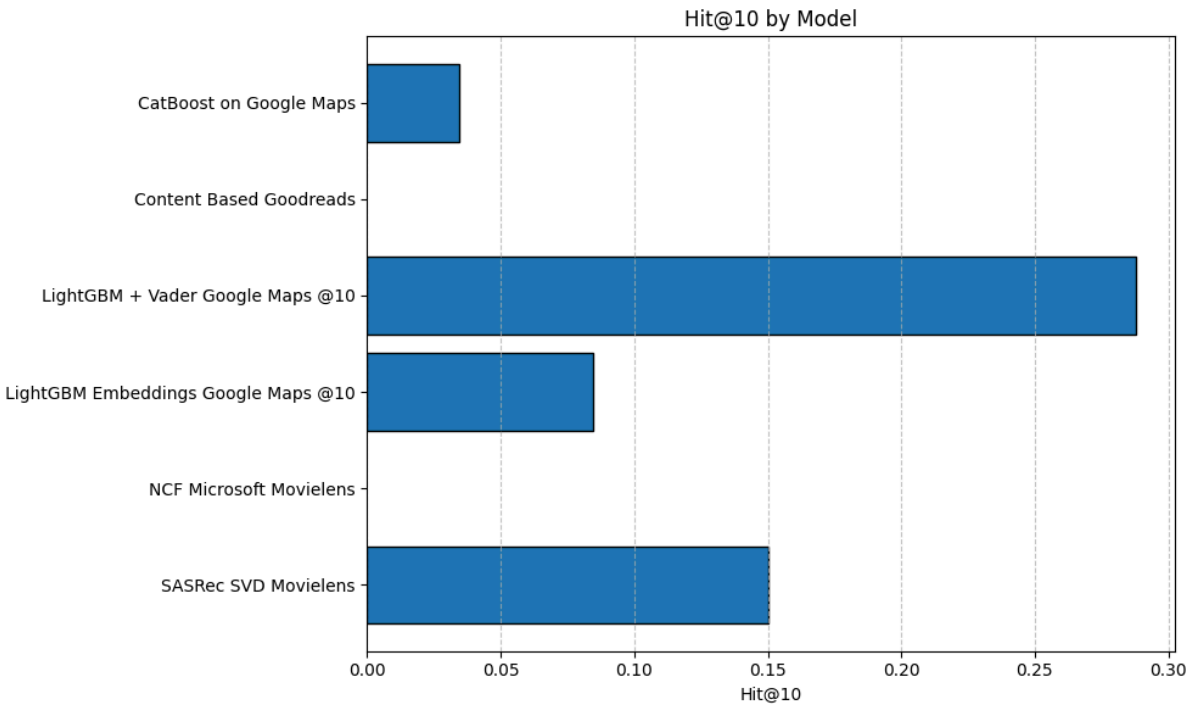
Graph Recall at 10



Graph NDCG at 10



Graph Map at 10



Graph Hit at 10