Final Report

DATA 1030

ScM CS - Minghao Liang

minghao_liang@brown.edu

GitHub Repo

1. Introduction

Problem Motivation

Online news recommendation systems are critical in enhancing user engagement and satisfaction. Effective recommendations not only enhance user experience but also increase advertising revenue and readership loyalty for the platform.

Data Overview

The dataset is an open-sourced dataset for a data science competition, extracted from a popular news app with a very high daily active user. The dataset samples approximately 4 minutes window of real click information from this app and includes:

- **Users:** Approximately 300,000 unique users.
- **Clicks:** Nearly 3 million user click events.
- **Articles:** About 360,000 news articles, accompanied by metadata and embedding vectors.

Problem Definition

The task is to predict the Top 5 news articles each user is likely to click on, leveraging historical user behavior and article information. This involves ranking a subset of articles for each user based on relevance.

Previous Work

Research on recommendation systems has been a popular area in academia. Traditional machine learning methods have been increasingly supplemented and enhanced by deep learning methods. However, traditional models still demonstrate strong performance. Considering the computational limitations and the complexity of deep learning models, which exceed the scope of this course, this project adopts traditional machine learning methods.

Since this project is based on a non-academic open-source dataset, there are no related academic papers tested on this dataset. Also, to maintain academic integrity, I didn't directly search and check other external pipelines and model designs specific. I only visited and learned limited and unverified information (primarily from open-source communities and forums), so far, the final model performance is comparable to or these references. The comparison with baseline performance further validates this point

2. EDA Analysis

Data Overview

In this dataset, all data has been masked, click information has been embedded and encoded in certain ways. Thus, the original data does not contain any semantically explicit -- it lacks rich user profile features and contextual features. Furthermore, the dataset does not include explicit reflection, it relying solely on implicit feedback through click behavior. This requires feature construction and sampling strategies to convert click logs into clear classification or ranking tasks. The dataset contains no missing value.

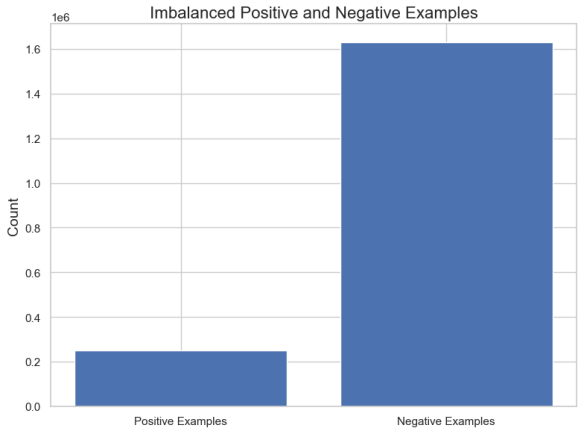

Figure.1 Distribution of positive and negative examples

Critical Features for Feature Engineering

User Preferences for Article Length:

User preferences for article length are evident. Figure 1 shows the distribution of news article word counts is fairly uniform, while actual user clicks show a pronounced long-tail effect in Figure 2. This insight led to features like word_count, min_word_count, and max_word_count related to word count.

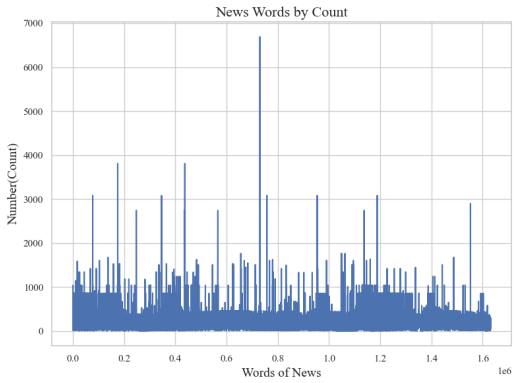

Figure.2 Distribution of News Word Count

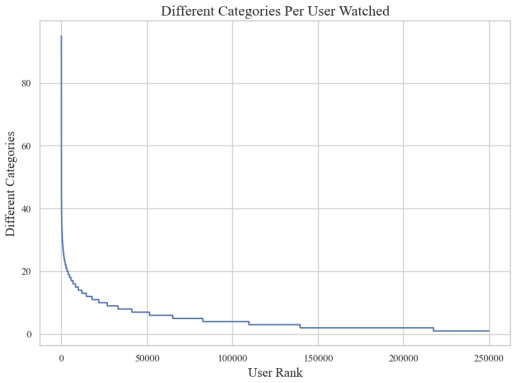

Figure.3 Categories Every User Watched

User Preference on Category:

The dataset contains articles across various categories, with some categories being more prevalent than others. Click distribution across categories varies, indicating differences in user interest. (Figure.3)

Figure. 4 demonstrates that users tend to click on news within the same category during a specific timeframe.

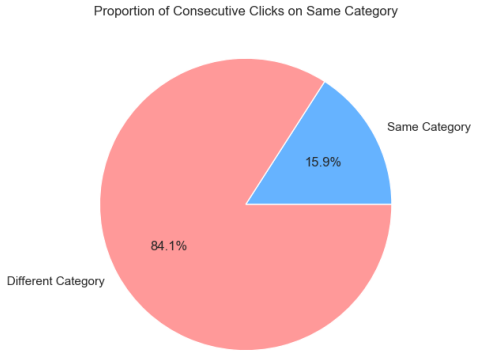

Figure.4 Proportion of Consecutive Clicks on Same Category

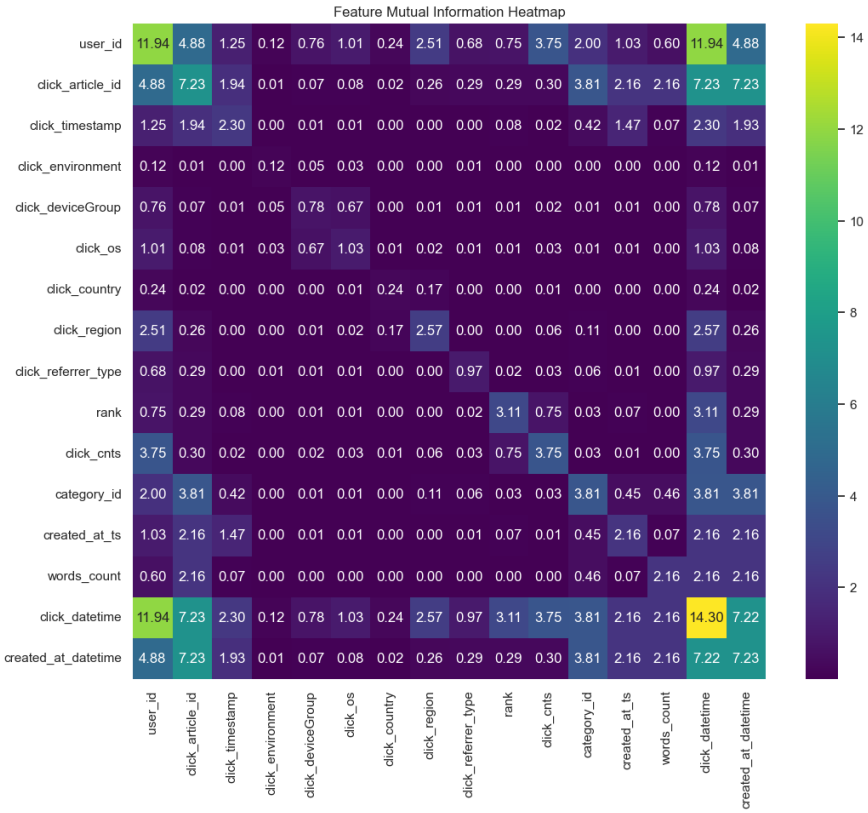

Figure.5 Feature Mutual Information Heatmap

Timestamps and User Activity Pattern:

1. The feature mutual information heatmap (Figure. 5) shows that time-related features always have interaction with other features.

2. The click distribution rises sharply in the 0–2 hour range, peaks at 3 hours, and then gradually declines, with a sharp drop-off after 27 hours. This trend reflects the strong timeliness of news articles and its influence on user click behavior.
3. The time intervals between consecutive user clicks also reflect click patterns.

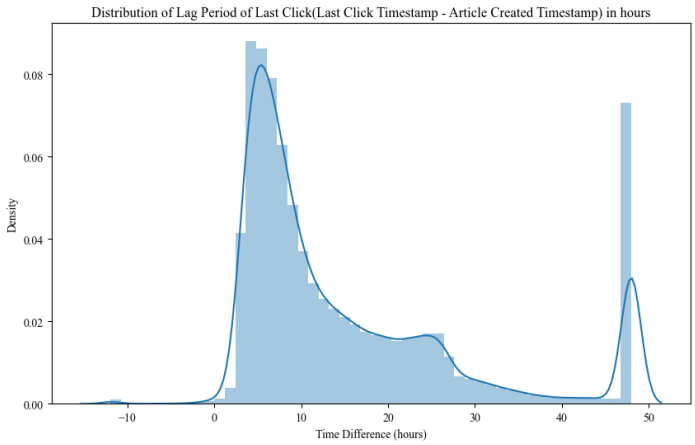

Figure.6 Distribution of Lag Period during Last Two Clicks

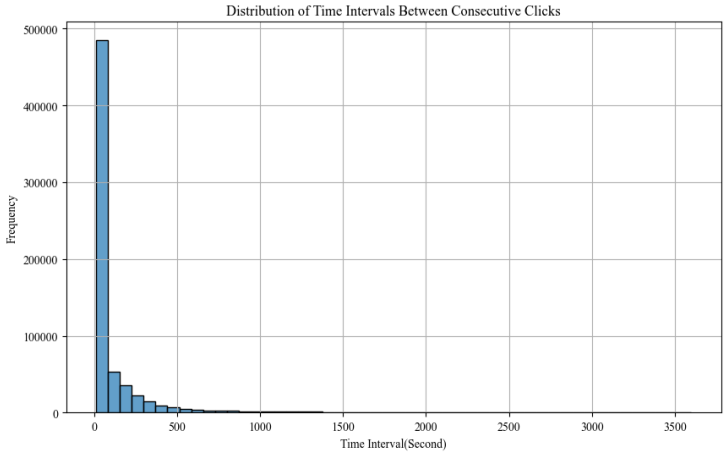

Figure.7 Distribution of Time Intervals between Consecutive Clicks

4. Methods

Data Preprocessing and Splitting:

GroupKFold cross-validation was employed to ensure user data did not overlap between training and validation sets, providing a realistic estimate of model

generalization. Specifically, user click sequences before the last click timestamp were used as the training set. For validation, the last click behavior was extracted from the training set as the validation user set, with subsequent clicks as positive examples for performance evaluation. Duplicate entries were removed, and invalid user IDs were filtered.

One important point to note is that I tested different random state parameters for various hyperparameter combinations. The results indicate that nearly all hyperparameter combinations remain consistent across different random state settings.

Feature Engineering:

Based on EDA insights, multi-dimensional features were manually extracted and constructed. These article attribute features, user-side features, similarity features constructed based on relationships between articles, and most importantly user behavior features based on TimeStamp (Figure.2). Key engineered features include:

1. Average, minimum, and maximum word count of articles viewed by the user.
2. Time interval between consecutive clicks and submission intervals between clicked articles.
3. Differences between the word count of the last viewed article and the user's average word count.
4. Time difference between the last click and article submission time.

Metrics

In terms of metric selection, this study uses Mean Reciprocal Rank (MRR) as the primary indicator:

$$score(user) = \sum_{k=1}^{5} \frac{s(\text{user}, k)}{k}$$

Where $s(user, k) = 1$ *if the prediction k for this user matches the user's last click data, otherwise* $s(user, k) = 0$

MRR is highly sensitive to the position of the correct result in the recommended list, accurately reflecting the model's sorting quality for the user's next click. MRR has an intuitive interpretation in this task: the earlier the correct recommendation, the higher the MRR. In addition, we also refer to HIT@K (exactly the same thing with Recall@5) as supplementary indicators. These additional indicators help to verify and explain the model's performance from more perspectives, ensuring that the model not only ranks the correct result higher but also maintains a high hit rate and relevance ranking performance within a certain range.

Cross Validation:

The recommendation system contains two stages. In the first stage, we use a clustering algorithm, Collaborative Filtering, for the first-step recall. In the second stage, we use a regression frame based model to score and rank the recalled news.

Since the data contains a large number of high-dimensional and non-linear relationships, we tried Logistic Regression as a baseline linear classifier. Non-linear models include Random Forest and Gradient Descent Decision Tree Models (such as LightGBM or XGBoost). Among them, LightGBM was chosen as a key candidate for the final model because it is faster and more memory-efficient, while potentially more accurate than XGBoost in some cases.

For parameter tuning and training details, we used Grid Search. During the tuning process, we searched for key hyperparameters of all models. Best parameter showed in Figure. 8

0.0.1 LGBMRanker Parameter Grid

Parameter	Values
n_estimators	[300, 500, 1000]
num_leaves	[31, 63, 127]
reg_alpha	[0.0, 0.1, 0.3, 0.5]
reg_lambda	[0.5, 1.0, 2.0, 5.0]

Parameter	Values
subsample	[0.5, 0.7, 0.9, 1.0]
colsample_bytree	[0.5, 0.7, 0.9, 1.0]
min_child_samples	[20, 50, 100]

0.0.2 Logistic Regression Parameter Grid

Parameter	Values
C	[0.001, 0.01, 0.1, 1, 10]
max_iter	[100, 200, 500]

0.0.3 Random Forest Parameter Grid

Parameter	Values
max_depth	[5, 10, 20, None]
random_state	[42, 777]

0.0.4 XGBoost Parameter Grid

Parameter	Values
max_depth	[3, 5, 7, 9]
min_child_weight	[1, 5, 10]
subsample	[0.5, 0.7, 1.0]
colsample_bytree	[0.5, 0.7, 1.0]

Model	Best Params	CV Avg MRR (±Std)	CV Avg HIT (±Std)	Test MRR	Test HIT
Baseline (ItemCF)	-	-	-	0.1222	0.4932
Logistic Regression	C=0.01, max_iter=100	0.2651 ± 0.0010	0.6007 ± 0.0023	0.2666	0.5984
Random Forest	max_depth=10, random_state=42	0.4084 ± 0.0016	0.6574 ± 0.0005	0.4055	0.6536
XGBRanker	max_depth=7, min_child_weight=5, subsample=0.7, colsample_bytree=1.0 random_state=42, early stopping = 10	0.4531 ± 0.0023	0.6657 ± 0.0016	0.4491	0.6588
LGBMRanken	_estimators−500, colsample_bytree=0.9, num_leaves=63. reg_alpha = 0, reg_lambda=2.0, subsample=0.7, early stopping = 10	0.4572 ± 0.0011	0.6647 ± 0.0004	0.4624	0.6657

Figure. 8 Model Performance Table

5. Results

We solely use the Top 5 news returned from the Item Collaborative Filtering recall model in the first stage without any further ranking process. The HIT of the baseline is 0.49 and the MRR of the baseline is 0.12. Since the baseline model uses one of the most efficient recall models in our project, it has a decent HIT score, but it cannot accurately identify the user's most interested news.

Aside from the baseline, we used two different Collaborative Filtering models in the first recall stage and used four non-linear regression models in the second rank stage. We compared multiple models and baseline methods, and ultimately selected the best-performing model for further analysis of feature importance and model interpretation.

The baseline model used only the top 5 news items returned by the Item-CF recall model without any ranking stage. On the test set, it achieved MRR=0.1222 and HIT=0.4932. While this baseline uses an efficient recall model, resulting in a decent HIT score, it lacked the ability to accurately prioritize the user's interests.

Introducing a ranking stage significantly improved performance. For example, using a Logistic Regression model yielded MRR=0.2666 and HIT=0.5984 on the test set, more than doubling MRR compared to the baseline, although there was still room for improvement.

Replacing the model with a nonlinear tree-based model (Random Forest) led to even better results, with MRR=0.4055 and HIT=0.6536 on the test set. This represented a

more than 200% increase in MRR over the baseline, though further refinement in handling complex, high-dimensional nonlinear features remained possible.

Adopting a ranking model (such as XGBRanker), which calculates scores for each data point within a group and sorts them accordingly, further enhanced performance. XGBRanker achieved MRR=0.4491 and HIT=0.6588 on the test set, demonstrating more stable and robust results. Ultimately, LGBMRanker delivered the best performance, with MRR=0.4624 and HIT=0.6657. Compared to the baseline (MRR=0.1222 and HIT=0.4932), this represented a remarkable improvement in overall recommendation quality.

Inteprebility

In this project, I used three global feature importance analysis methods, including Global Feature Importance (Gain) from LGBMRanker's feature importance evaluation (Figure. 9), Permutation Importance (Figure. 10), and SHAP global importance analysis (Figure.11). From the results, time-related features are the core of prediction performance, with *lag_period_last_article* and the *lag_between_created_at_ts_and_clicked_ts* both ranked as the most important features by all three methods. This indicates that the user's time behavior patterns play a key role in predicting user interest. Meanwhile, content features are equally important, with *words_count* and *category_id* both having high global importance, indicating that users' content preferencessignificantly affect the recommendation results.

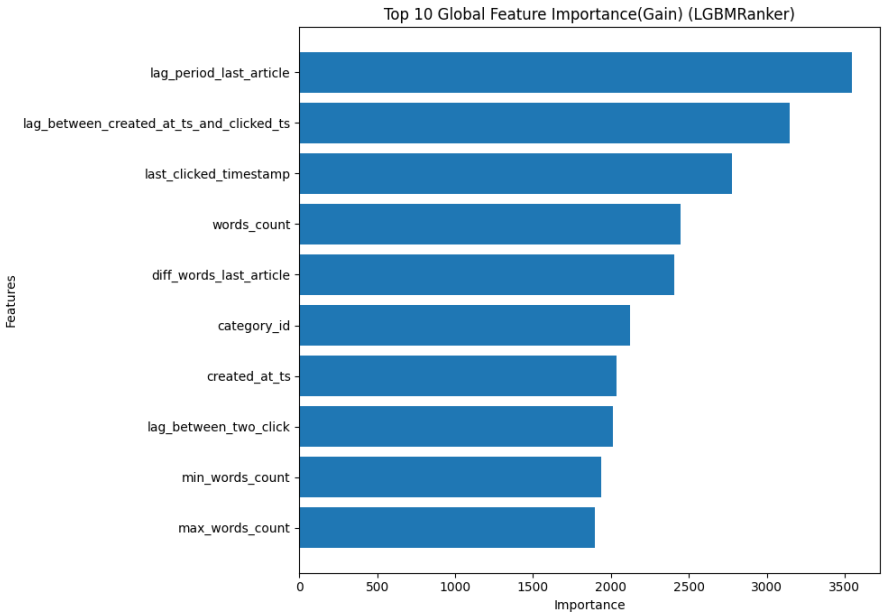

Figure. 9 Global Feature Importance(Gain) of LGBMRanker

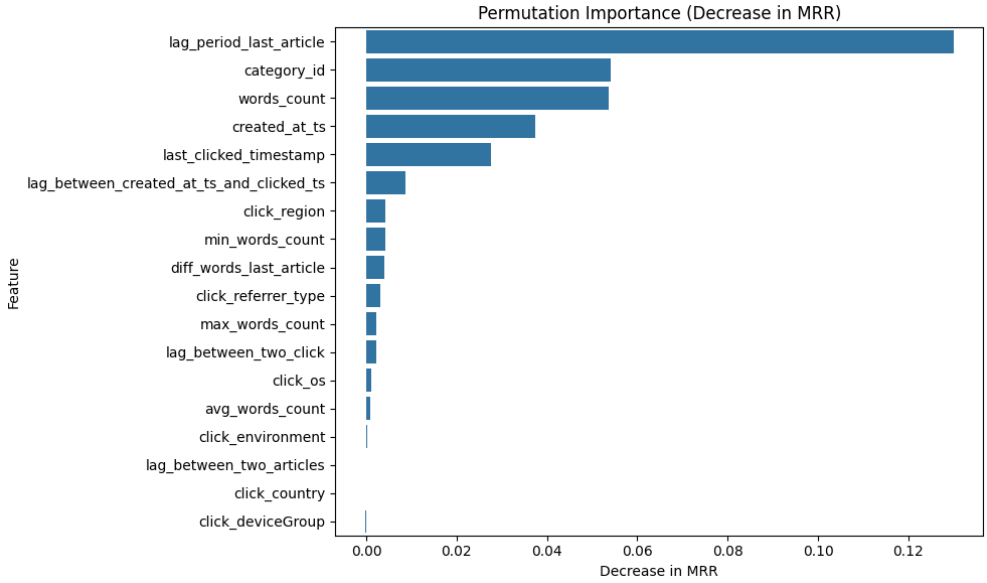

Figure. 10 Permutation Importance

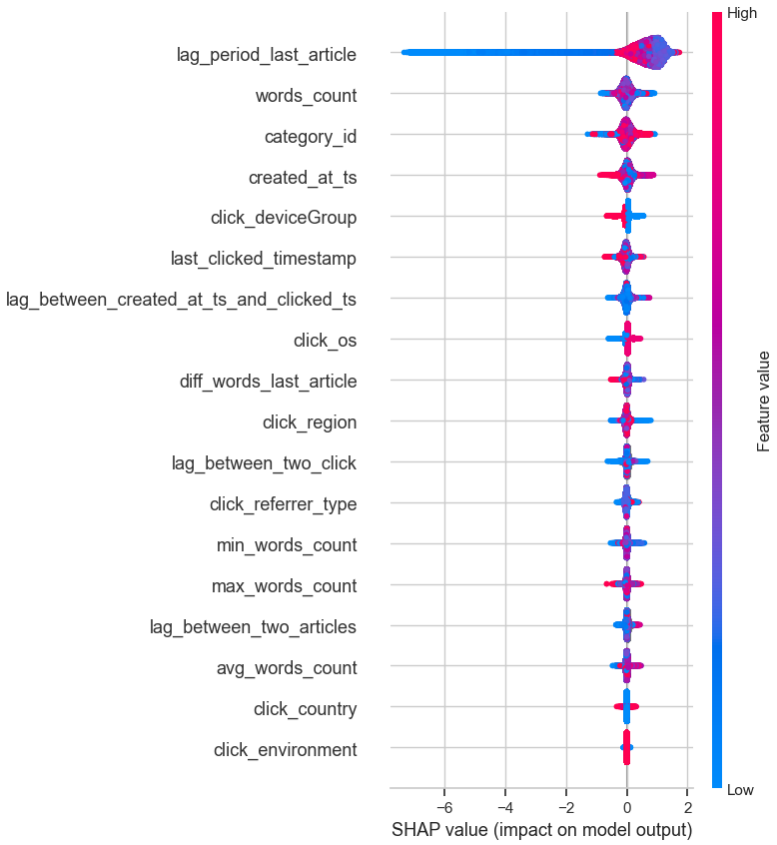

Figure. 11 SHAP Global Importance Analysis

By analyzing the SHAP values for the 0th, 500th, 5000th, and 10,000th users, we can see that time-related features always play a core role in recommendations for different users, reflecting the criticality of the time distribution of user behavior on recommendations. Content features are also valuable for capturing user preferences, especially when user interests are relatively stable. Environmental features, although showing specific influences for certain users, appear weaker overall and may require further exploration combined with personalized strategies.

Figure. 12 SHAP Value at Different Data Point

Based on our previous analysis and insights, we can summarize the potential logic behind users' clicking behavior:

• Users have a clear preference for certain article lengths and tend to click on articles that match their historical reading habits.

•Users often exhibit concentrated activity during specific periods, with shorter click intervals suggesting a focus on related topics.

•Users' clicks are significantly influenced by the publication time of articles, with many preferring recent content over older pieces.

•While users generally follow their historical preferences, certain events or special content may cause temporary deviations from their usual behavior.

•Users' click behavior can be personalized based on their historical reading habits

6. Outlook

Looking ahead, there are still many directions for further improvement in this study. First, at the model optimization level, in addition to the currently used tree-based models and traditional machine learning methods, future iterations could consider introducing more advanced and complex recall and ranking models, including deep learning recall frameworks similar to YouTube DNN, and neural network models such as DIN (Deep Interest Network) that can dynamically weight features based on the user's specific behavior sequence. Such models are expected to better capture the hierarchical structure of user preferences and temporal dependencies, achieving higher predictive accuracy and robustness. In addition, performing feature selection and dimensionality reduction on existing features, or using data augmentation to enrich the training set, may potentially help further improve model generalization performance.

On the data side, if more complete user behavior profiles can be collected, such as longer-term click sequences, more dimensions of user features, and more detailed geographic and temporal contextual information, the model will be provided with more in-depth and broad feature foundations. More detailed data not only improves the model's predictive accuracy but may also make the model more adaptable across multiple scenarios.

Finally, in terms of model interpretability and understandability, it is possible to use causal inference combined with randomized controlled trials (RCT) to experimentally analyze the true impact of recommending certain items on user behaviors (such as clicks and purchases), as well as differences in responses to recommendations among

different user groups. This helps design more refined recommendation strategies.

(Word Counts: 1946 words)

Reference

1. **https://tianchi.aliyun.com/competition/entrance/531842/information?lang=en-us**
2. **Wang SL, Wu CY. Application of context-aware and personalized recommendation to implement an adaptive ubiquitous learning system. Expert Syst Appl. 2011;38:10831–8.**
3. **Huang Z, Lu X, Duan H. Context-aware recommendation using rough set model and collaborative filtering. Artif Intell Rev. 2011;35:85–99.**
4. **Bala Krishnamoorthy and Rishi Jala**, "Build a news recommender application with Amazon Personalize," *AWS Machine Learning Blog*, April 04, 2024, [no page number], [Advanced (300), Amazon Personalize, Technical How-to], [Comments/Share section noted], **https://aws.amazon.com/cn/blogs/machine-learning/build-a-news-recommender-application-with-amazon-personalize/**
5. **Amal**, "AI-Powered News Article Recommendation System," *ScrapeHero* (published on Medium), September 12, 2023, [no page number], **https://medium.com/scrapehero/ai-powered-news-article-recommendation-system-1877fd423a80**
6. **Adam Cierlik**, "News Recommendation: the challenging area in building recommendation systems," *getindata.com Blog*, February 28, 2024, [no page number], **https://getindata.com/blog/news-recommendation-challenging-area/**
7. Xei, "Build a recommendation system with TensorFlow and Keras," GitHub, 2021, https://github.com/xei/recommender-system-tutorial
8. Webisoft, "How to Build Recommendation System: A Comprehensive Guide," Webisoft, 2023, https://webisoft.com/articles/build-recommendation-system/
9. Codez Up, "End-to-End Tutorial on Building a Recommendation System," Codez Up, 2023, https://codezup.com/recommendation-system-end-to-end-tutorial/
10. Stratoflow, "How to Build a Recommendation System: Explained Step by Step," Stratoflow Blog, 2023, https://stratoflow.com/how-to-build-recommendation-system/
11. Unite.AI, "Building a Recommendation System Using Machine Learning," Unite.AI, 2022, https://www.unite.ai/building-recommendation-system-using-machine-learning/
12. Machine Learning Mastery, "A Practical Guide to Building Recommender Systems," Machine Learning Mastery, 2024, https://machinelearningmastery.com/practical-guide-building-recommender-systems/

13. NVIDIA Developer Blog, "How to Build a Deep Learning Powered Recommender System, Part 2," NVIDIA Developer Blog, 2020, https://developer.nvidia.com/blog/how-to-build-a-winning-recommendation-system-part-2-deep-learning-for-recommender-systems/
14. Nick McCullum, "Recommendation Systems in Python - A Step-by-Step Guide," Nick McCullum, 2023, https://www.nickmccullum.com/python-machine-learning/recommendation-systems-python/
15. SpringerLink, "How to Build a Recommendation System for Software Engineering," SpringerLink, 2015, https://link.springer.com/chapter/10.1007/978-3-319-28406-4_1
16. SabrePC, "Tutorial: Recommender System using TensorFlow & TFRS," SabrePC, 2023, https://www.sabrepc.com/blog/Deep-Learning-and-AI/How-to-Build-a-Recommender-System-using-TensorFlow