International Journal of Core Engineering & Management
Volume-7, Issue-11, 2024 ISSN No: 2348-9510

ENHANCING RECOMMENDATION SYSTEMS WITH ADVANCED FEATURE ENGINEERING

Pushkar Mehendale
Drift.com
San Francisco, CA, USA
pushkar.mehendale@yahoo.com

Abstract

Recommendation systems are essential in various domains, from e-commerce to streaming services, by predicting user preferences and providing personalized suggestions. However, the effectiveness of these systems heavily depends on the quality of features used in the models. This paper explores advanced feature engineering techniques to enhance recommendation engines, focusing on integrating content-based, collaborative filtering, and hybrid approaches. We review state-of-the-art methods and propose a framework for feature engineering that incorporates deep learning and contextual information to improve recommendation accuracy and address issues such as the cold-start problem. Special emphasis is given to the choice of LightFM for modeling and the solutions implemented at Drift to address cold-start challenges. This paper aims to provide insights into the latest advancements in feature engineering for recommendation systems and their practical applications.

Keywords: Recommendation Systems, Feature Engineering, Cold-Start Problem, LightFM.

I. INTRODUCTION

Recommendation systems (RS) play a critical role in modern online platforms by helping users navigate large volumes of content and discover items of interest. Traditional RS methods, such as collaborative filtering and content-based filtering, have been widely adopted but face challenges [3], [8], especially with sparse data and cold-start problems [9], [10]. This paper aims to enhance the performance of RS by leveraging advanced feature engineering techniques. We will explore how combining different data types can lead to more accurate and robust recommendations. RS have become essential components of modern online platforms, assisting users in navigating vast content libraries and discovering items that align with their preferences. Traditional RS methods, such as collaborative filtering and content-based filtering, have demonstrated their effectiveness in various applications. However, they often encounter challenges when dealing with sparse data, where user-item interactions are limited, and cold-start problems, where new users or items have little or no interaction history.

To address these challenges, this paper proposes enhancing the performance of RS by leveraging advanced feature engineering techniques. Feature engineering is the process of transforming raw data into features that are more informative and predictive. By combining different data types and applying appropriate feature engineering techniques, we aim to extract more meaningful insights from the available data and improve the accuracy and robustness of recommendations.

One key aspect of our approach is the integration of user and item metadata. User metadata, such as demographics, location, and browsing history, can provide valuable insights into user preferences and behaviors. Similarly, item metadata, such as content, category, and tags, can capture the inherent characteristics of items. By combining these metadata sources, we can construct richer feature representations that better reflect the relationships between users and items.

Additionally, we explore the use of advanced feature engineering techniques to extract more complex and discriminative features. For instance, we employ natural language processing (NLP) techniques to analyze item content and extract information. We also leverage matrix factorization and graph-based methods to capture latent factors and structural relationships within the user-item interaction data. These advanced feature engineering techniques allow us to uncover hidden patterns and relationships that traditional methods may overlook.

By combining different data types and employing advanced feature engineering techniques, we aim to create a more comprehensive and expressive feature space that better represents the preferences and interactions of users and items. These enhanced features can then be utilized by RS algorithms to generate more accurate and personalized recommendations.

II. LITERATURE REVIEW

1.1 Collaborative Filtering

Collaborative filtering (CF) is a popular method for building recommender systems (RS). It utilizes historical interaction data between users and items, such as ratings, purchases, or views, to make predictions about a user's preferences. CF methods can be categorized into two main approaches: user-based CF and item-based CF.

User-based CF identifies users with similar tastes and preferences based on their past interactions with items. Once similar users are identified, the system recommends items that those similar users have enjoyed but the target user has not yet interacted with. This approach assumes that users with similar tastes in the past are likely to have similar tastes in the future [3].

Item-based CF, on the other hand, identifies items that are similar to items that a user has previously interacted with. This is achieved by analyzing the historical interactions of all users with different items. Once similar items are found, the system recommends them to the target user. The assumption behind this approach is that items that are similar in terms of their features or content are likely to be appreciated by the same users [8].

1.2 Content-Based Filtering

Content-based filtering (CBF) is a recommendation technique that leverages item features to suggest similar items to users. Unlike collaborative filtering, which relies on user-item interactions, CBF focuses on item characteristics. This approach is particularly useful in scenarios where user interaction data is limited, such as when a user is new to a platform or when there is a lack of sufficient historical data.

CBF works by extracting features from items, such as product descriptions, genres, and other attributes. These features are then used to create a representation of each item. When a user interacts with an item, CBF recommends similar items based on the similarity between their feature representations. The effectiveness of CBF depends on the quality and comprehensiveness of the item metadata. If the metadata is sparse or inaccurate, the recommendations may not be as relevant or personalized. However, when the metadata is rich and well-structured, CBF can provide highly accurate and personalized recommendations. Additionally, CBF can be combined with other recommendation techniques, such as collaborative filtering, to further improve the quality of recommendations.

III. Hybrid Approaches

Hybrid recommendation systems are a versatile and effective way to leverage the strengths of both collaborative filtering (CF) and content-based filtering (CBF) methods. By combining the two approaches, hybrid systems can mitigate the weaknesses of each individual method and provide more accurate and personalized recommendations [2].

One way to implement a hybrid recommendation system is to simply combine the predictions from CF and CBF models. This can be done by taking a weighted average of the two predictions, or by using a more sophisticated method such as a stacking ensemble. Another approach is to use CBF to enhance CF with additional features. For example, CBF can be used to extract features from user-generated content, such as reviews or ratings, and these features can then be used to improve the performance of a CF model.

Finally, deep learning can be used to learn complex user-item interactions in a hybrid recommendation system. Deep learning models can be used to extract features from user-item interactions, and these features can then be used to train a CF or CBF model. Deep learning can also be used to build a hybrid model that combines the strengths of both CF and CBF.

Hybrid recommendation systems are a powerful tool for providing accurate and personalized recommendations. By combining the strengths of CF and CBF, hybrid systems can overcome the limitations of each individual method and provide a more comprehensive and effective recommendation service.

IV. ADVANCED FEATURE ENGINEERING TECHNIQUES

4.1 Deep Learning for Feature Extraction

Deep learning models, such as neural networks, convolutional neural networks (CNNs), and recurrent neural networks (RNNs), have revolutionized the field of feature extraction. These models have the ability to learn complex relationships within data, allowing them to extract high-quality features that are essential for various machine learning tasks [2].

For example, CNNs have proven to be highly effective in extracting features from images and videos. Their ability to recognize patterns and identify objects within visual data has made them indispensable in applications such as image classification, object detection, and facial recognition. Additionally, RNNs excel at extracting features from sequential data, such as user browsing history or natural language text. Their ability to capture temporal relationships and dependencies within data sequences has made them invaluable for tasks like machine translation, sentiment analysis, and time-series forecasting.

The success of deep learning models in feature extraction can be attributed to their ability to learn hierarchical representations of data. These models can automatically discover and learn important features at different levels of abstraction, starting from low-level features (like edges and corners in images) to high-level features (like objects and scenes). This hierarchical representation allows deep learning models to capture both local and global patterns within the data, resulting in rich and informative features.

4.2 Contextual and Temporal Features

Incorporating contextual information into recommendation systems can greatly improve their accuracy and effectiveness. Contextual information includes factors such as time of day, location, and user mood. By taking these factors into account, recommendation systems can provide more personalized and relevant recommendations. For example, a music streaming service might recommend different songs to a user depending on the time of day or their current location. Similarly, a news app might recommend different articles to a user based on their mood.

Temporal features, which capture the timing of interactions, can be particularly useful in modeling user preferences more dynamically. For example, a user's preferences for movies might change depending on the time of day or day of the week. Techniques such as time-aware collaborative filtering and contextual bandits are effective in integrating these features into recommendation systems. Time-aware collaborative filtering takes into account the time of day or day of the week when a user interacts with the system, while contextual bandits allow the system to learn from user feedback over time and adapt its recommendations accordingly.

V. Hybrid Feature Engineering

Feature engineering is a crucial step in building robust recommendation models. By combining features from multiple data sources, such as user profiles, item attributes, and interaction data, models can better understand the complex relationships between users and items. Feature crossing, embedding layers, and attention mechanisms are effective techniques for capturing these interactions.

Feature crossing involves combining different features to create new features that represent more complex relationships. For example, crossing a user's age with their gender can create a new feature that represents their age group and gender combination. Embedding layers are used to represent categorical features as dense vectors, which allows models to learn the similarities and differences between different categories. Attention mechanisms enable models to focus on specific parts of the input data, such as particular items or features, when making recommendations [7].

These techniques help recommendation models capture the nuanced relationships between users and items, resulting in more accurate and personalized recommendations. By combining features from multiple data sources and applying feature engineering techniques, models can better understand user preferences and item characteristics, leading to improved recommendation performance [4].

VI. LIGHTFM FOR ENHANCED RECOMMENDATIONS

6.1 Overview of LightFM

LightFM, a widely used recommendation library, leverages a potent combination of collaborative and content-based filtering approaches through matrix factorization. This unique blend enables LightFM to effectively handle various types of feedback data, including implicit (e.g., clicks, views) and explicit (e.g., ratings, purchases). Moreover, LightFM excels in sparse datasets, a common challenge in recommendation systems. In such scenarios, where user-item interactions are limited, LightFM's ability to make accurate predictions sets it apart [1].

LightFM operates by learning embeddings for both users and items. These embeddings are essentially low-dimensional representations that capture the essential characteristics of each user and item. The embeddings are then utilized to estimate user preferences based on their historical interactions and the metadata associated with the items. This approach allows LightFM to make personalized recommendations that align with individual user tastes and preferences.

Furthermore, LightFM's versatility extends to its applicability across domains. It has been successfully employed in diverse domains, including e-commerce, streaming services, and social media platforms. In these scenarios, LightFM has demonstrated its ability to enhance user engagement and satisfaction by providing highly relevant and personalized recommendations.

6.2 Advantages of LighFM

LightFM's ability to handle hybrid recommendations is a significant advantage, as it can combine information from both user-item interactions and item features to generate more accurate recommendations. This is particularly useful in situations where there are a limited number of user-item interactions, as it can leverage the additional information provided by item features to improve the quality of recommendations.

LightFM also supports a variety of loss functions, including WARP (Weighted Approximate Rank Pairwise), BPR (Bayesian Personalized Ranking), and logistic loss. This allows users to choose the loss function that best suits their specific needs and data characteristics. For example, WARP is often used for ranking-based recommendation tasks, while BPR is better suited for top-N recommendation tasks [1].

6.3 Applications at Drift

LightFM's flexibility and robustness made it an ideal choice for Drift's recommendation engine. LightFM can handle both user and item metadata, which allowed Drift to create a more comprehensive model that addressed the diverse needs of users. By incorporating various types of data, Drift was able to provide more relevant and personalized recommendations. Additionally, LightFM's efficient handling of sparse data was crucial for mitigating the cold-start problem, which is a common challenge in recommender systems. This ability ensured that Drift could make accurate recommendations even for users or items with limited interaction history.

LightFM's flexibility enabled Drift to customize the recommendation engine to specific business requirements. For example, Drift was able to incorporate domain-specific knowledge and business rules into the model. This customization allowed Drift to create a recommendation engine that was tailored to the unique needs of its users. Additionally, LightFM's robustness ensured that the recommendation engine was able to handle large amounts of data and perform well under varying conditions. This robustness was essential for Drift to provide a reliable and scalable recommendation service.

Overall, LightFM's flexibility, robustness, and efficient handling of sparse data made it an ideal choice for Drift's recommendation engine. These features allowed Drift to create a comprehensive and accurate recommendation engine that met the diverse needs of its users.

V. ADDRESSING THE COLD START PROBLEM

5.1 Challenges at Drift

The cold-start problem is a prevalent challenge in recommendation systems, particularly for new users or items with minimal or no interaction data. This issue arises when a recommendation algorithm has limited information to draw upon, making it challenging to provide accurate and personalized recommendations. At Drift, we encountered this problem firsthand while aiming to offer tailored recommendations to new website visitors and recently added content.

To address the cold-start problem, we implemented several strategies. One approach involved leveraging user demographics and contextual information, such as the user's location, device type, and browsing history. By incorporating these additional data points, we were able to make more informed recommendations, even for users with limited interaction history. Additionally, we utilized collaborative filtering techniques to identify similarities between new users and existing users with similar preferences. This allowed us to recommend items that had been well-received by users with comparable profiles.

We employed active learning methods to gather additional information from new users. By presenting users with a small number of carefully selected items, we were able to elicit feedback and preferences, which we then used to refine our recommendations. This iterative process allowed us to learn more about new users over time, enabling us to provide increasingly personalized recommendations.

5.2 Item Embeddings Integration

To address the cold-start problem, which arises when new items with limited interaction data are introduced, we incorporated item embeddings as features into our LightFM collaborative filtering model. Item embeddings enable us to represent items in a high-dimensional space based on their attributes and features. This representation allows the model to capture the underlying relationships between items, even if they have not yet accumulated a significant number of interactions with users. By utilizing these embeddings, our model can make more informed and accurate recommendations for new items, even with limited interaction data.

The process of integrating item embeddings into the LightFM model involved clustering similar items based on their attributes and features. This clustering allowed us to group together items that share common characteristics, such as genre, category, or brand. Once the items were clustered, we used the resulting clusters as additional features in our LightFM model. By incorporating these cluster features, the model could leverage the relationships between items within each cluster to make recommendations for new items, even if they had not yet been interacted with by many users.

This approach proved to be effective in mitigating the cold-start problem and enhancing the overall performance of our recommendation system. By utilizing item embeddings, we were able to provide more diverse and relevant recommendations for new items, ultimately improving the user experience and driving engagement with the platform [11].

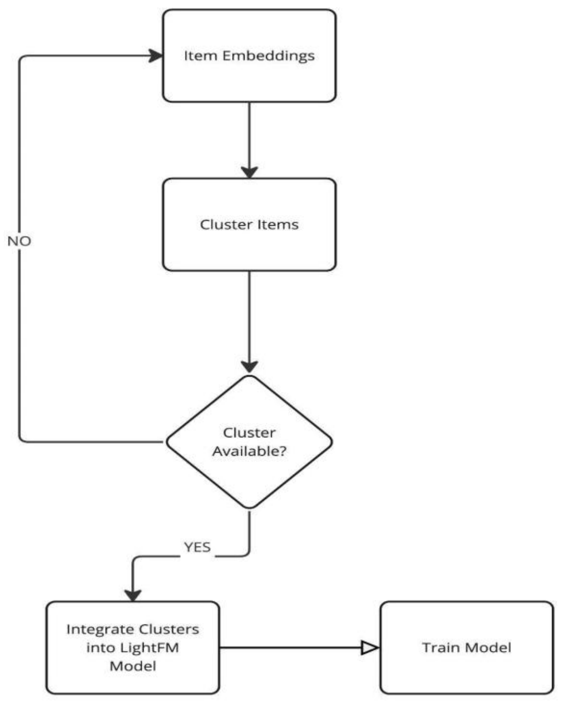

Figure 1: Item Embeddings Integration Workflow

5.3 Clustering Techniques

In order to group comparable items based on their embeddings, we applied clustering methods like K-means and DBSCAN. With this strategy, we were able to establish informative clusters that the recommendation system could leverage to deduce user preferences for novel products [5]. By optimizing clustering parameters and using dimensionality reduction techniques like PCA, we were able to increase the effectiveness and precision of our clustering procedure [6], [10].

K-means and DBSCAN are two clustering algorithms that are well-suited for this task. K-means divides the data into a specified number of clusters, while DBSCAN finds clusters of arbitrary shape and size. By using these algorithms together, we were able to create clusters that were both meaningful and efficient.

We further refined our clustering process by optimizing the clustering parameters and using dimensionality reduction techniques. By tuning the parameters of the K-means and DBSCAN algorithms, we were able to improve the quality of the clusters. Additionally, we used PCA to reduce the dimensionality of the data, which made the clustering process more efficient.

5.4 Global and Custom Clustering

In order to improve our model, we incorporated both global and custom organization-specific clusters. Global clustering allowed us to maintain a broad understanding of item similarities across different contexts. This approach ensured that our recommendations were generalizable to a wide range of users and situations. For instance, if a user from one organization expressed a preference for a particular item, our model could recommend similar items to users from other organizations, even if those users had different preferences or contexts.

Custom clusters, on the other hand, enabled us to tailor recommendations to specific organizational needs. This approach allowed us to take into account the unique characteristics of each organization, such as its industry, size, and culture. By creating custom clusters for each organization, we could ensure that our recommendations were relevant to the specific needs of that organization's users. For example, a healthcare organization might have different recommendation needs than a financial services organization.

The combination of global and custom clustering allowed us to create a recommendation system that was both generalizable and contextually relevant. This dual approach ensured that our recommendations were useful to users from a variety of organizations and contexts [11], [12].

VII. CONCLUSIONS

Recommendation engines play a vital role in enhancing user experiences across various online platforms. However, traditional recommendation systems often face limitations due to their inability to capture complex user preferences and handle data sparsity effectively. Advanced feature engineering techniques offer a promising solution to address these challenges and improve recommendation accuracy and robustness.

One key aspect of advanced feature engineering is the integration of deep learning models. These models can extract meaningful representations from raw data, capturing intricate patterns and relationships that traditional methods may overlook. By leveraging deep learning techniques, recommendation engines can provide more personalized recommendations tailored to individual user preferences. Additionally, contextual features, such as user location, time, and device type, can be incorporated to enhance the relevance and timeliness of recommendations.

Hybrid approaches, which combine traditional recommendation methods with advanced feature engineering techniques, offer a powerful way to address the limitations of both approaches. For example, LightFM, a state-of-the-art recommendation engine used at Drift, incorporates deep learning and contextual features to provide accurate and personalized recommendations. Strategies such as addressing the cold-start problem, which occurs when new users or items have limited interaction data, are crucial for the success of recommendation systems. By leveraging advanced feature engineering techniques, we can effectively mitigate this issue and provide meaningful recommendations for new users and items.

VII. ACKNOWLEDGEMENT

We extend our gratitude to the Drift machine learning team for their invaluable contributions to this project and the insightful feedback from our peer reviewers.

REFERENCES

1. Kula, Maciej. 2015. “LightFM: A Python Implementation of a Hybrid Recommender System.” ArXiv abs/1507.08439.
2. Guo, Huifeng, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. “DeepFM: A Factorization-Machine-Based Neural Network for CTR Prediction.” In Proceedings of the 26th International Joint Conference on Artificial Intelligence, 1725–31. Melbourne, Australia: AAAI Press.
3. Nilashi, Mehrbakhsh, Othman Ibrahim and Karamollah Bagherifard. “A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques.” Expert Syst. Appl. 92 (2018): 507-520.
4. Chen, Qiwei, Huan Zhao, Wei Li, Pipei Huang and Wenwu Ou. “Behavior sequence transformer for e-commerce recommendation in Alibaba.” Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data (2019).
5. Rama, Kiran, Pradeep Kumar and Bharat Bhasker. “DNNRec: A novel deep learning based hybrid recommender system.” Expert Syst. Appl. 144 (2020): 113054.
6. Wang, Kai, Tiantian Zhang, Tianqiao Xue, Yu Lu and Sang-Gyun Na. “E-commerce personalized recommendation analysis by deeply-learned clustering.” J. Vis. Commun. Image Represent. 71 (2020): 102735.
7. Wang, Hao, Shan You, Xianzhi Wang, Zheng Liu, Huawei Shen, Chang Xu, Xueqi Cheng, and Liangjie Hong. 2022. “A Survey on Session-Based Recommender Systems: Taxonomy and Future Directions.” ACM Computing Surveys 55 (9): 1–35.
8. Patoulia, Agori Argyro, Athanasios Kiourtis, Argyro Mavrogiorgou and Dimosthenis Kyriazis. “A Comparative Study of Collaborative Filtering in Product Recommendation.” Emerging Science Journal (2022).
9. Huang, Feiran, Zefan Wang, Xiao Huang, Yu-hong Qian, Zhetao Li and Hao Chen. “Aligning Distillation For Cold-start Item Recommendation.” Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (2023).

10. Kannout, Eyad, Marek Grzegorowski, Michal Grodzki and Hung Son Nguyen. "Clustering-Based Frequent Pattern Mining Framework for Solving Cold-Start Problem in Recommender Systems." IEEE Access 12 (2024): 13678-13698.
11. Siet, Sophort, Sony Peng, Sadriddinov Ilkhomjon, Misun Kang and Doo-Soon Park. "Enhancing Sequence Movie Recommendation System Using Deep Learning and KMeans." Applied Sciences (2024).
12. Li, Zhen and Wang, Jibin and Chen, Zhuo and Wu, Kun and Liu, Liang and Ai, Meng and Liu, Li. "Drm4rec: A Doubly Robust Matching Approach for Recommender System Evaluation" (2024).