Engineering Scalable Recommender Systems: From Model Selection to E-Commerce Implementation

Soël MEGDOUD
Candidate

Valentin GORCE
Supervisor

Thomas TAYLOR
Supervisor

Abstract

This paper presents the development of recommendation system pipelines for large-scale e-commerce data. Two pipelines were designed: a personalized recommendation system and an item-similarity pipeline, both optimized for inference at scale using MLOps best practices.

A method to label CRM [1] data, based on chronological splitting and negative sampling allowed to provide realistic training conditions for e-commerce applications without data leakage.

The DLRM[2] recommender system from Meta achieved the best performance, while simpler models such as Random Forest showed competitive results. These architectures have demonstrated their effectiveness in handling an extensive dataset, enabling them to deliver recommendations in near real-time.

1 Introduction

This report details the work conducted during my internship at Headmind Partners, a consulting firm specializing Digital, Cybersecurity and AI. As part of the AIOps department, I was tasked with developing advanced recommender systems. Using the H&M CRM dataset from a Kaggle competition, which includes extensive customer and product information for about 1M customers and 500k products, I developed two pipelines: a personalized recommendation system and an item-similarity recommendation system. The goal was to create scalable, real-time recommendation systems applicable to e-commerce scenarios, following MLOps best practices and using state-of-the-art models. To further validate the system's performance in real-world conditions, a demostrator simulating an e-commerce website was developed.

[1] Customer Relationship Management software
[2] Deep Learning Recommender System

2 Pipelines architectures

2.1 Personalised recommendations pipeline

The personalized recommendation pipeline utilizes a multi-stage architecture designed to efficiently return relevant item lists for users.

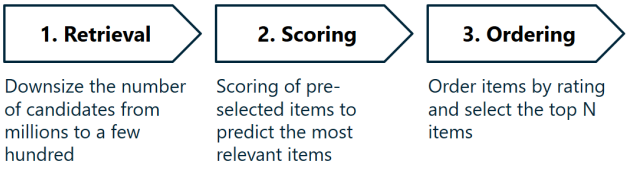

Figure 1: Multi-stages recommendation pipeline

Given a user ID, its row features are retrieved from a feature store (Feast) and processed in-line with Nvtabular, which maps categorical attributes to integers and normalizes numerical values.

For retrieval, a Two-Tower model facilitates fast retrieval of candidate items by leveraging pre-computed item embeddings. The user-embedding is calculated using the Query-Tower and to enhance retrieval speed, we implement a FAISS index, which clusters item embeddings and uses approximate vector search techniques to quickly identify the nearest items embeddings.

Subsequently, candidate item features are also fetched in the feature store and processed for scoring.

Finally, the processed features of both users and candidate items are fed into a scoring model, such as DRLM or Wide&Deep, to rank the items and select the top N recommendations. This systematic approach ensures an efficient and accurate personalized recommendation process.

2.2 Item Similarity Pipeline

The item similarity pipeline recommends products similar to the one a user is currently viewing, providing a content-based recommendation system. By retrieving the embedding of the consulted item and comparing it with other embeddings using FAISS, the pipeline returns a list of similar items. For this purpose, FashionCLIP, a fine-tuned version of the CLIP model, is used to generate precise product embeddings.

3 Evaluation and results

3.1 Creation of a Labeled Dataset

To train the scoring models, a labeled dataset was created using H&M's CRM data, which consists of customer purchase histories. The dataset was split chronologically to avoid future data leakage. Purchases from the last 20 days were used as the positive interactions, while earlier data formed the training set.

Since there are no explicit negative interactions (i.e., products viewed but not purchased), a sampling method was employed to generate them. Negative examples were generated by sampling random customers and items with a distribution indexed on recent customers and very popular products to avoid relying solely on popularity. This approach also ensures an equal balance of positive and negative samples to help the model to focus on the products purchased (which reflects the user's real interests).

3.2 Model Performance

In order to evaluate different scoring models, I tested both tree-based models, like Random Forest and Gradient Boosting, as well as deep learning models, including DLRM, Wide&Deep, DCN, and a simple Multi-Layer Perceptron (MLP). The evaluation was based on three metrics: AUC [3], Precision, and Recall.

Model	AUC	Precision	Recall
Random Forest	0.8050	**0.8326**	0.7488
Gradient Boosting	0.7981	0.8226	0.7440
DLRM	**0.8794**	0.8236	0.7100
Wide&Deep	0.8485	0.8104	0.6416
DCN	0.8286	0.7569	0.6814
MLP	0.8195	0.7072	**0.7543**

Table 1: Scoring models performance

The DLRM model achieved the highest AUC, demonstrating superior performance in ranking relevant items compared to other models. Tree-based models, such as Random Forest and Gradient Boosting, showed strong precision and recall, with Random Forest achieving the highest precision. However, deep learning models like DLRM and Wide&Deep still performed comparably in terms of Precision.

The simple MLP model, lacking pairwise feature interaction, exhibited lower precision, which underscores the importance of modeling feature interactions in recommendation systems.

3.3 Inference Times

The operational performance of the recommendation pipeline was also evaluated, with a focus on inference times.

- Customized recommendation pipeline: The inference time is approximately **1.4 seconds**, whereas asynchronous inference averages **1 second**
- Item similarity pipeline: The inference time is around **0.3 seconds**, which is significantly faster as the pipeline is less complex.

Most of the inference time is spent on processing steps, involving the Nvtabular workflows and processing raw data from the feature store. These steps account for approximately 1.2 second of the total 1.4 seconds. On the other hand, model retrieval and inference times are minimal due to preloading into RAM at server startup.

4 Conclusion and Perspectives

The architectures developed have proven to be effective for handling a large-scale dataset and enabling near real-time inference. Incorporating a retrieval step was essential in managing a vast number of items.

Optimizations, such as storing processed data in the feature store, could drastically reduce the inference time. However, taking feature processing out of the pipeline makes maintenance and addition of new items and customers much more complex. There is a trade-off between inference speed and operational complexity.

A notable achievement was the implementation of a method for generating a labeled dataset from CRM data, providing a more realistic evaluation of model performance.

State-of-the-art models performed well in terms of AUC. However, the evaluation metric, AUC, while useful, does not fully capture the quality of the ranked recommendations. Other metrics such as nDCG [4] or MAP [5] could have provided more insights into the ranking performance. Furthermore, the system was only tested in offline environments, and no A/B testing was conducted in a real-world production setting.

[3] Area Under the Curve

[4] Normalized Discounted Cumulative Gain

[5] Mean Average Precision