[ML-6889] Recommender Systems with LLMs

Background

To improve user engagement and content discovery, we identified the need for personalized action link recommendations on the homepage and global search results. These recommendations help surface relevant items tailored to each user's interests and behavior, enhancing the overall user experience. By integrating an LLM-based recommendation engine, we aim to generate context-aware suggestions that beyond keyword matching and supports both cold start and return user scenario.

Motivation

Below is the list of pain points of existing global search results:

- Many user queries are partial and fragmented, making it difficult to identify consistent intent patterns. This underscores the need for auto-complete or auto-suggestion functionality to assist users in formulating complete and meaningful queries. We believe that complete queries are key to encourage more open-ended and exploratory searches, as it allows us to better interpret user intents and serve results across the breadth of content available.
- New users and actions lack sufficient interaction history, making it difficult to generate relevant recommendations. Meanwhile, the current system relies heavily on click signals, which are really sparse or unavailable in most scenario, especially when user click history is missing or weak.
- Action links meta data such as captions and descriptions are available, but there is no robust content-based recommendation pipeline leveraging metadata or contextual attributes for recommendations, instead of purely relying on clicked action popularity.

Given that, business team proposed the following functional for recommender systems:

- Persona-Based Recommendations on the first Login
    - When a user logs in for the first time, the system must:

      • Identify the user's persona based on available user profile data (role-based: HRP, Payroll Practitioner, Manager, etc.).

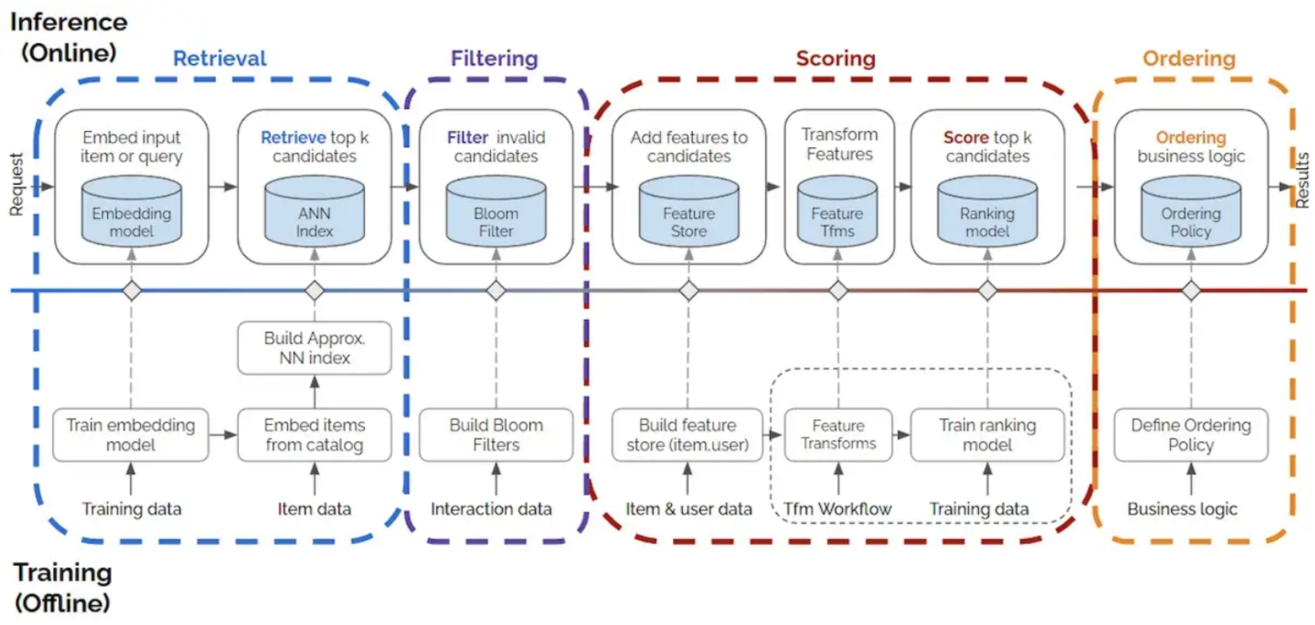
      • Immediately provide relevant, personalized content or action recommendations aligned with their identified persona.

      • This is specifically targeting users for whom we do not have any usage data and are unable to provide personalized recommendations.
- Dynamic Recommendations Based on Historical Interactions

In this document, a technical design is provided to provide an overview and high-level POC about how the feature could be implemented. Take cold start and content-based model as examples.

Recommender Examples

How do the architecture for industrial search and recommendation system looks like? There are handful of papers and references that discuss implementation details elucidate design patterns and best practices. These works helped inspired KDD 2022 to host a dedicated workshop for Recommender System. In a keynote, Even Oldridge and Karl Byleen-Higley from Nvidia presented the following recommendation system architecture:



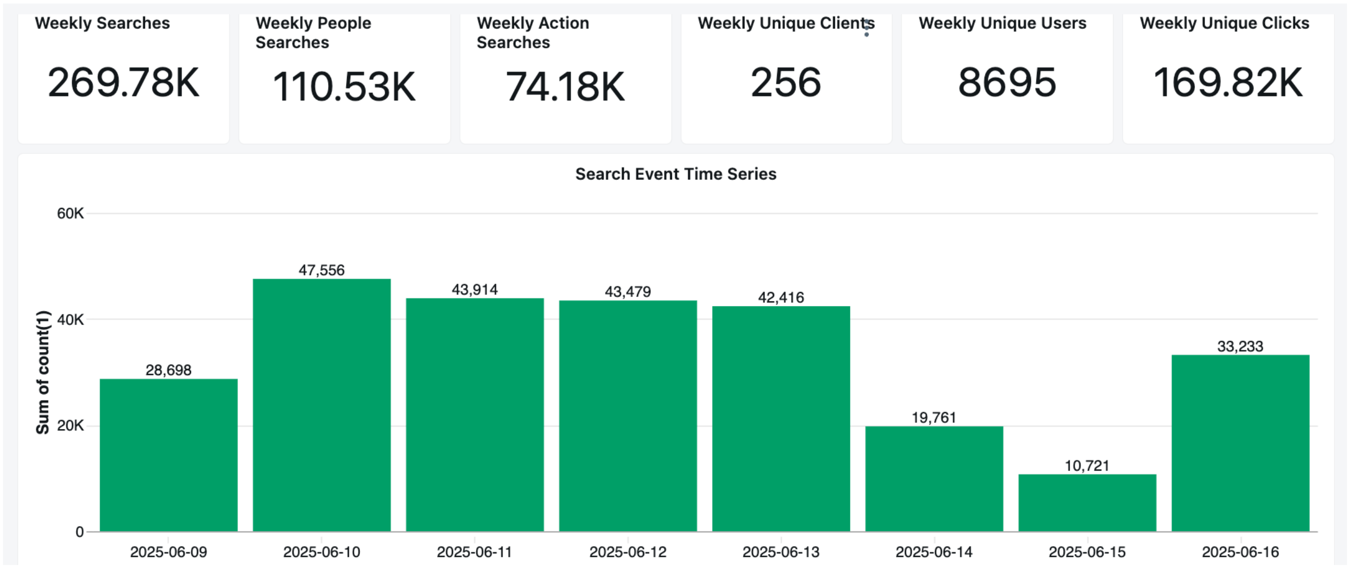
Even Oldridge & Karl Byleen-Higley (NVIDIA) augmented my 2-stage design with 2 more stages.

The offline environment largely hosts batch processes such as creating features or embeddings for items, building an approximate nearest neighbors (ANN) index or knowledge graph to find similar items, and model training (e.g., ranking). It may also include loading item and user data into a feature store that is used to augment input data during ranking.

The online environment then uses the artifacts generated (e.g., ANN indices, knowledge graphs, models, feature stores) to serve individual requests. A typical approach is converting the input item or search query into an embedding, followed by candidate retrieval and ranking. There are also other preprocessing steps (e.g., standardizing queries, tokenization, spell check) and post-processing steps (e.g., filtering undesirable items, business logic, etc).

Establish Design Scope

The prod system needs to support approximately 300 clients real time, handling a weekly load of around 270K search queries, with a peak of ~50K on a weekday and ~170K click events. There are about 111K weekly active users, and 30K daily active users. We know that the usages aren't evenly distributed, that is, peak activity occurs on weekdays, with little or even no data available on weekends.



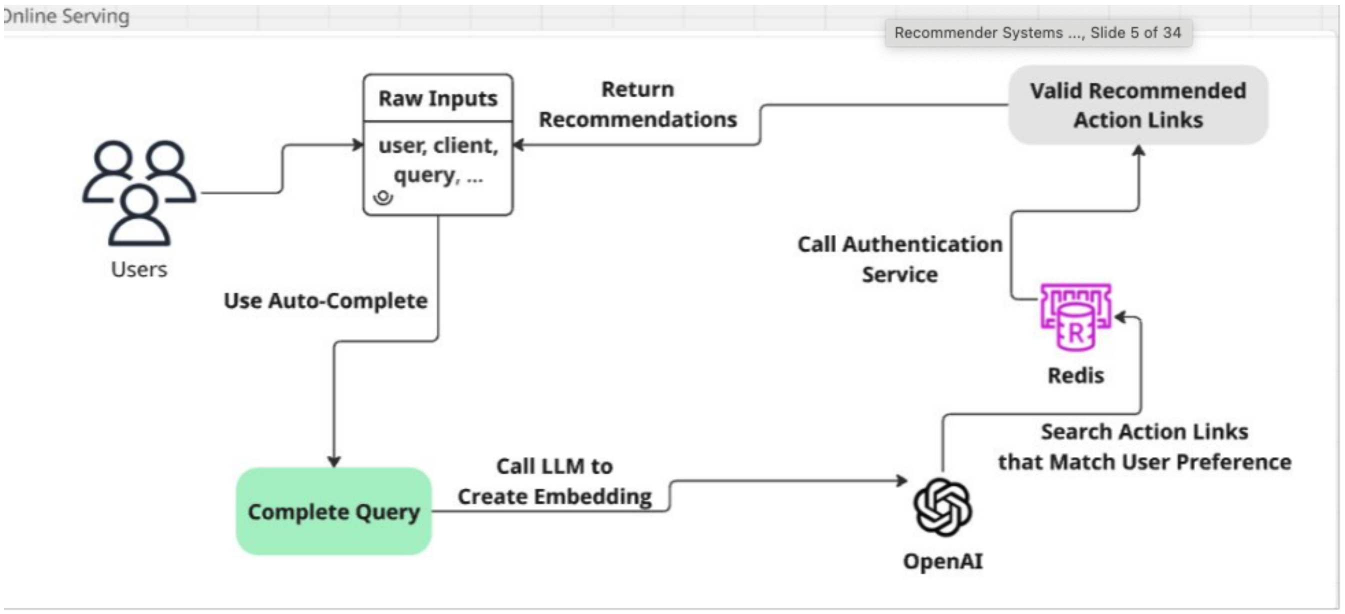
Weekly Searches	Weekly People Searches	Weekly Action Searches	Weekly Unique Clients	Weekly Unique Users	Weekly Unique Clicks
269.78K	110.53K	74.18K	256	8695	169.82K

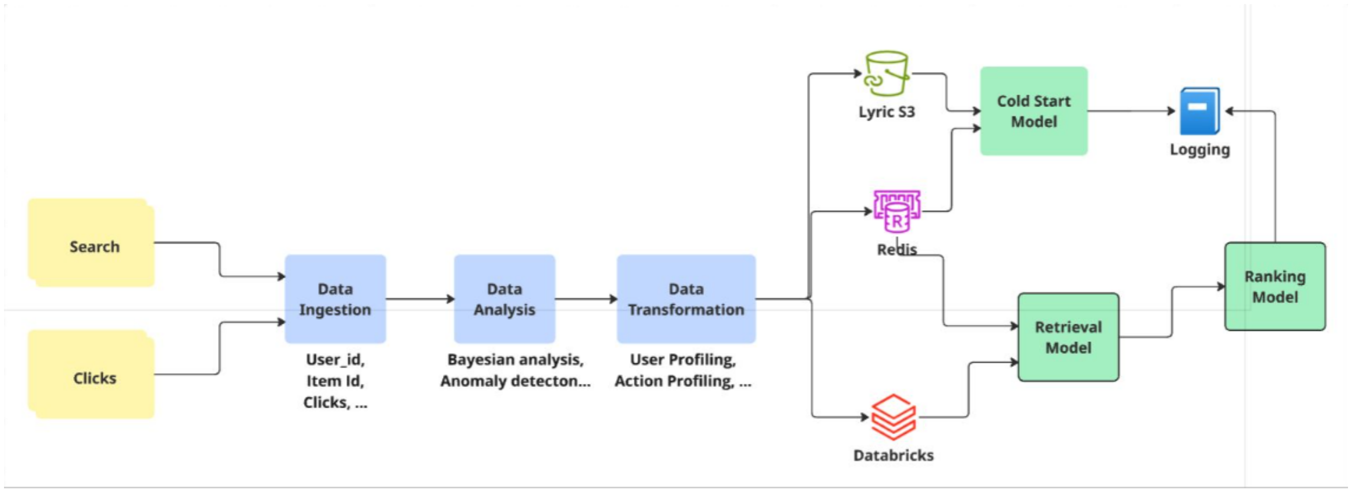
Search Event Time Series

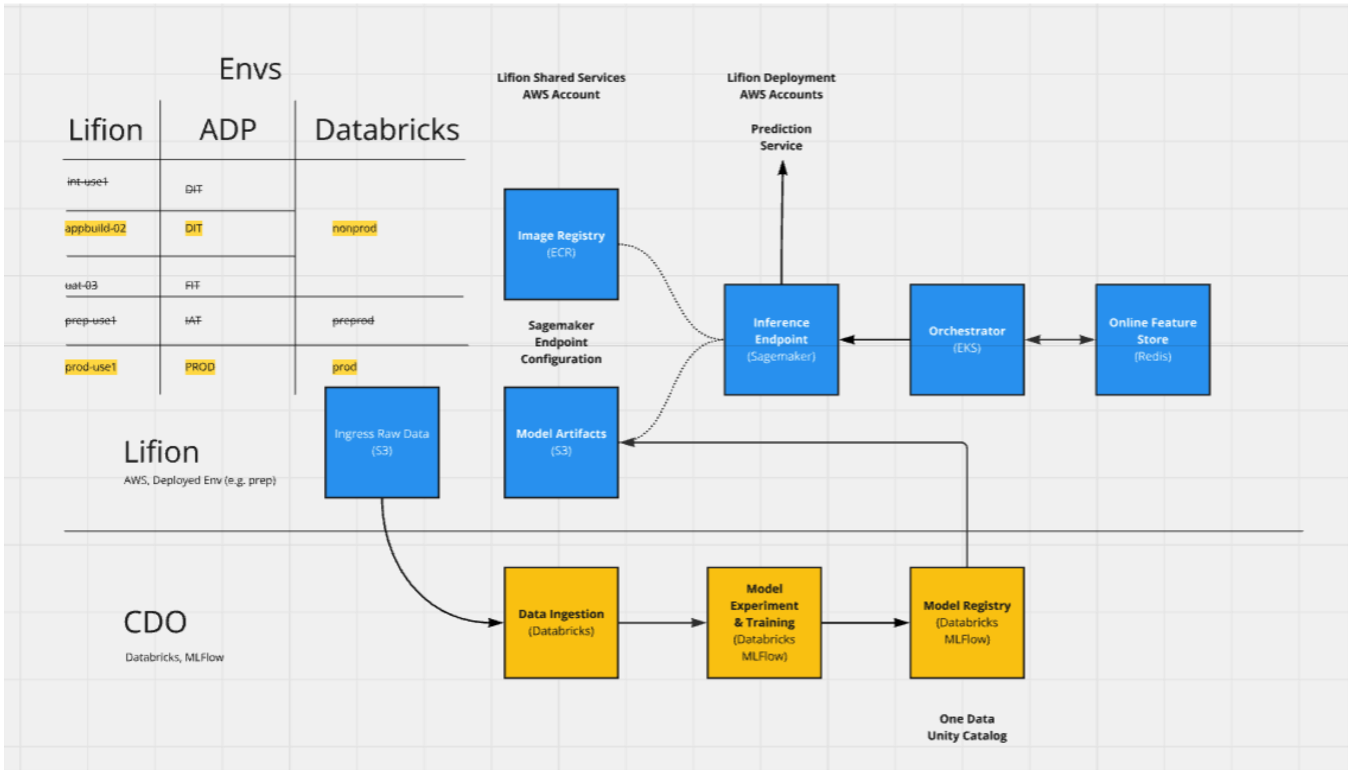
	Sum of count(1)
2025-06-09	28,698
2025-06-10	47,556
2025-06-11	43,914
2025-06-12	43,479
2025-06-13	42,416
2025-06-14	19,761
2025-06-15	10,721
2025-06-16	33,233

Given the average throughput equal to 900 per minute, we would like to have an average of 15 users per second. If we assume that peak load would be 5 times the average, the QPS for users is about 75.

High Level Design


Online Serving
Recommender Systems ..., Slide 5 of 34
Users
Raw Inputs
user, client, query, ...
Return Recommendations
Valid Recommended Action Links
Use Auto-Complete
Call Authentication Service
Redis
Search Action Links that Match User Preference
Complete Query
Call LLM to Create Embedding
OpenAI




Envs

Lifion	ADP	Databricks
~~int-use1~~	~~DIT~~	
appbuild-02	DIT	nonprod
~~uat-03~~	~~FIT~~	
~~prep-use1~~	~~IAT~~	~~preprod~~
prod-use1	PROD	prod

Lifion Shared Services AWS Account
Lifion Deployment AWS Accounts
Prediction Service
Image Registry (ECR)
Sagemaker Endpoint Configuration
Inference Endpoint (Sagemaker)
Orchestrator (EKS)
Online Feature Store (Redis)
Lifion
AWS, Deployed Env (e.g. prep)
Ingress Raw Data (S3)
Model Artifacts (S3)
CDO
Databricks, MLFlow
Data Ingestion (Databricks)
Model Experiment & Training (Databricks MLFlow)
Model Registry (Databricks MLFlow)
One Data Unity Catalog