Product Roulette Design

Requirement : Dynamic recommendation system for B2B products based on user category

Basic details :

1. **Prediction** : Could be broken down into two categories as follows —>
   A. Existing User :
      - Top N Based on the collaborative filtering for the user
      - Top N best rated in the given category (to incorporate the item range issue)
      - Few picks from the new products in the category

   B. New User :
      - Top N best rated in the given category
      - Few picks from the new products in the category

** Margin of new products and best rated ones to be decided as per business.

2. Feedback Collection :
For second and further logins of a user, show the list of recommendations in the previous login and ask to choose 3 relevant items and rate them on a scale of 1 to 5

** Rating part included later on to make the process quantitatively more accurate as number of hits would not be be that good a stat.

3. Derivations :

A. user_id -> integer number generated as hash of mail_id
B. product_id -> integer number generated as hash of product_id

4. Process pointers :

A. Data needs to be already computed and present so that at run-time the API just hits the table to fetch results for a user/user-category
B. Initially till a certain cap on users/ feedbacks, batch processing to be done every few hours
C. Later on, frequent updates at category levels in parallel

5. Computations :

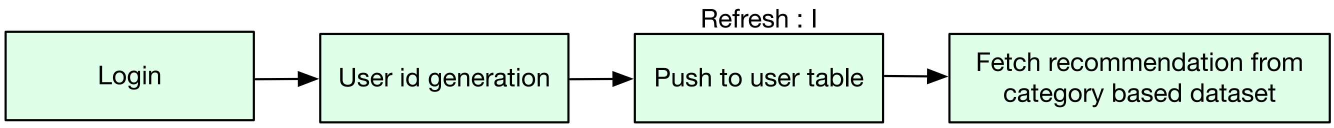
A. For existing users —>
   - Similarity of items in a category
   - Prediction for individual product entries for users

B. For new users —>
   - Depends on item ratings in the category
   - To incorporate number of ratings for a product, weighted rating to be used

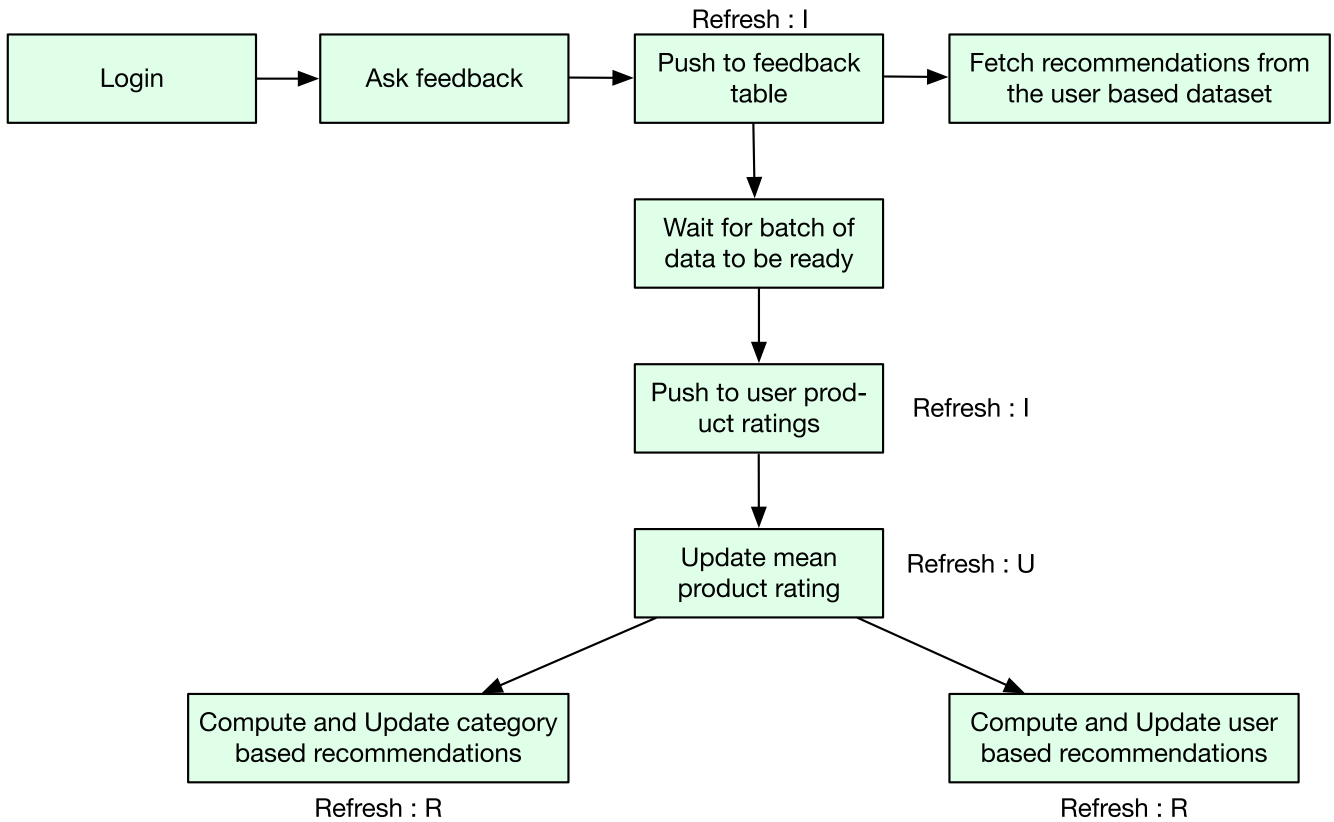
Flow Diagram

Broken down pipes :

First login :



Re-login :



Computation details :

1. User based —>
   - Content based causes major bias and also new(different) products are usually never recommended
   - Collaborative filtering is a better option
   - Issues and resolution in the current approach :
     - New user :
       - Recommendations to be made based on best products in the user category
     - New product :
       - New products to be included as a small percentage of overall products recommended at any time
   - User-user filtering not to be preferred as it included heavy computations, can recommend a given product even in case of very less evidence(ratings)
   - Item-item on the other hand will be more stable and more data driven result
   - The issue of conservative recommendations could be handled by recommending top products also as a percentage

Item-item similarity to be computed first based on user-ratings

$$w_{ij} = \mathrm{sim}(i,j) = \frac{\sum_{u \in U_i \cap U_j} \hat{r}_{ui}\hat{r}_{uj}}{\sqrt{\sum \hat{r}_{uj}}\sqrt{\sum \hat{r}_{uj}}}$$

Prediction to be computed based on rating given by user and corresponding rating for similar products.

$$s(i;u) = \mu_i + \frac{\sum_{j \in I_u} (r_{uj} - \mu_j) w_{ij}}{\sum_{j \in I_u} |w_{ij}|}$$

2. New user(content) based ->
   - Top products in each category can be recommended based on user
   - The issue here is whether to prefer a product with 1*5 star rating or 10*4 star ratings
   - This could be handled with weighted rating where a mean rating term could be added to handle fluctuations due to small set of user ratings

weighted rating (WR) = (v ÷ (v+m)) × R + (m ÷ (v+m)) × C

where:

R = average for the product (mean)
v = number of ratings for the product
m = tuneable parameter, ratings to be added with average rating to smoothen the result-set(more significant initially)
and ratings don't fluctuate the average drastically
C = the mean rating across the whole product set

Reference : https://stackoverflow.com/questions/1411199/what-is-a-better-way-to-sort-by-a-5-star-rating

Database selection :

- Initially for smaller data-sets, database is not needed as such.
- Data could be saved in csv files and processed using adhoc clusters
- While scaling, OLTP based database will be needed as there are many data update scenarios in the pipeline and need to be fast
- Redshift(more on OLAP side though) or Aurora seem to be two options
- Partitioning also needed so as to process the data in parallel for different categories
- Based on the current dataset, the number of dimensions are very limited and hence even with more incoming data, we are good to go with Aurora having a upper limit of 64TB

Data processing :

- Initially batch processing for the data using lambda function / airflow cron job
- Later on, for real time pipeline, Kinesis could be used to induct the data and micro-batching for running the updates and other computations using a run-time spark based processing cluster

Tables needed :

1. Two target tables :
   a) User level recommendations
   b) Category level recommendations

2. Dimension tables :
   a) User table
   b) Product table
   c) User product rating table
   d) Similarity results table

3. Feedback related ones :
  a) Feedback forward pipe -- collecting user rating, prod name and user identifier -- same as 2c.
  b) Feedback reverse pipe -- Backup of user level for previous login

4. Process related ones :
  a) Timestamp of data pulled in till last run

Table schemas :

1.a User level recommendations —>
fields : user_id
product_name
product_manufacturer
product_rating
semantic_ts
category (partition column)

1.b Category level recommendations —>
fields : product_name
product_manufacturer
product_rating
semantic_ts
category (partition column)

2.a User table —>
fields : user_id
user_mail_id
user_company
category (partition column)

2.b Product table —>
fields : product_id
product_name
product_manufacturer
mean_product_rating
no_of_ratings
category (partition column)

2.c User product rating table —>
fileds : user_id
product_id
user_rating
semantic_ts
category (partition column)

2.d Similarity results table
--> fields : current_product_id
related_product_id
similarity_score
category (partition column)

3.a Feedback forward pipe(reduntant) —>
fields : user_id
product_id
user_rating
timestamp

3.b Feedback reverse pipe —> same as 1.a (preferably subset as only product names are needed at user level)

4.a Process related ones —>
fields : category_name
interface_name
last_processed_ts