Documentation

Problem Statement

Given the transactional history of food ordered in restaurants in airports, provide two types of recommendations. General recommendation for home page on basis of time metadata and Add-to-Cart recommendation for when a user clicks on a certain item.

Data Source

The data was obtained from OTG itself. The data was stored in Amazon Redshift and extracted using pyscopg2 in Python.

Queries

```python
# For only one vendor
query = f"SELECT item_id, order_id, itm_unit_price FROM
otg_analytics.fct_ord_item_dtl WHERE vendor_id={vendor_id}"

# For unique items in that vendor
query = f"SELECT item_id, item_name, unit_price, product_type, r
FROM otg_analytics.dim_item WHERE item_id in
{unique_items_string}"

# For unique items in that vendor
query = f"SELECT menu_desc, item_ctg3_desc, item_ctg2_desc,
item_ctg_desc, item_id, is_alcohol_fl, is_item, is_modifie FROM
otg_analytics.dim_item_hrc WHERE item_id in
{unique_items_string}"

# For only one vendor
query = f"SELECT item_id, order_id, itm_unit_price FROM
otg_analytics.fct_ord_dtl WHERE vendor_id={vendor_id}"
```

Fields

1. Order_id : Unique order identifier

2. Item_id: Unique item identifier
3. Item_name: Name of the item
4. Unit_price: Price of the item
5. Item_ctg_desc: Description of the item
6. Item_ctg2_desc: Second level description of the item (more specific)
7. Item_ctg3_desc: Third level description of the item (more specific, generally null)
8. Ord_beg_time: Order begin time (extracted hour & weekdays)
9. Is_alchohol_fl: Is item alcohol
10. Is_item: Is item a standalone item
11. Is_modifer: Is item a modifier to a standalone item

Data Details

Data Range

Two years worth of transactional data of one vendor.

Number of Data

Around 10 million transactions.

Data Transformations

- Items which were modifier (is_modifier) were filtered out
- Since only one row of `item_ctg_desc` was missing, it was filled manually
- There was only item of class `Starbucks Na` in `item_ctg_desc`, it was replaced with `Non Alcoholic`
- Missing value of `item_ctg2_desc` were filled with mode of `item_ctg2_desc` grouped by `item_ctg_desc`
- Filtered items were removed from orders
- Length of each order was calculated (number of items purchased in one go), and order of length 1 was discarded
- All of the categorical variables were converted to indices for embedding

Analysis

Exploratory Data Analysis (EDA)

Single Vendor EDA After Integrating Weather Data

EDA Slides

Some of the important Insights are:

- There are some clear popular items in both food and drink categories.
- Egg & Styles and drip coffee and coke with flatiron burger and club sandwich are most popular items in purchase combinations.
- Food items and drinks preference can be distinguished based on temperature , humidity and precipitation.

EDA for Multiple Vendors

This was done to identify the common menu items across vendors from different geographical areas. First, all the airports were divided into five geographical areas and five biggest vendors in terms of the number of menu items for each geographical area were selected.

The items available for vendors were found to be clearly different for each geographical area.
EDA for Multiple Vendors

Modeling

A target for this problem was to generate recommendations using DeepFM algorithms which can extract high-level items and user information. There were two types of models being built: General recommendation and Add-to-Cart recommendation. There were not any significant developments for General Recommendation. Add-to-Cart recommendation was being built using sequential recommendation algorithms like SR-GNN, SASRecF, NextItNet and one DeepFM model DSIN.

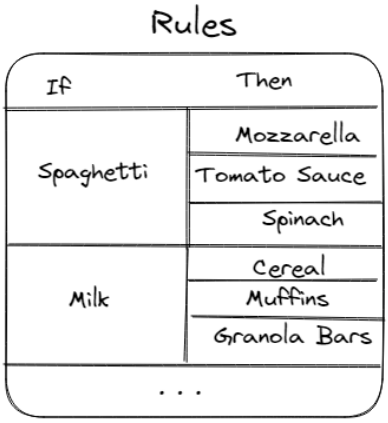
Algorithm Overview

Association Rule Mining based recommendation (FP-Growth)

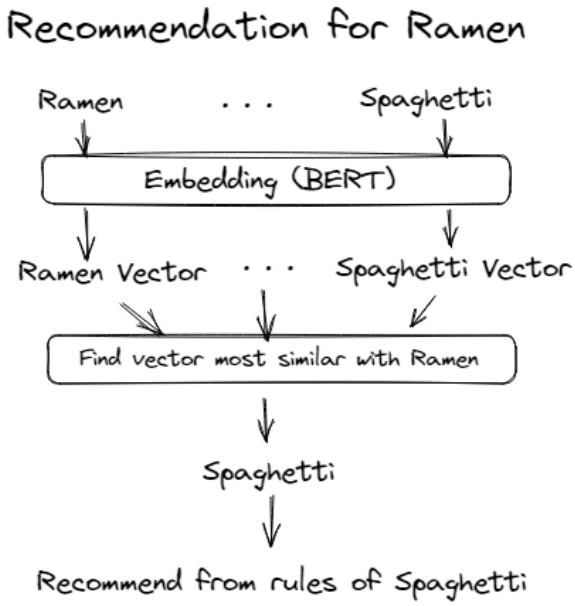
FP-Growth is a popular alternative to Apriori Growth algorithm which can be used to find frequently occurring itemset in a database. FP-Growth is faster than Apriori since it only needs to scan the database twice. The algorithm for recommendation is describe below:

- Find frequent item sets from the database with minimum confidence (Minimum probability of item B being purchased given item A has been purchased) and minimum

support (Minimum percentage of number of times item A and B have been purchased together). The frequent itemsets are called rules.



- If the product in the cart is in rules generated by FP-Growth, recommend those items.
- If the product in the cart is not in rules generated by FP-Growth, recommend using rules of most similar items. (Similarity computed using embedding generated from pretrained models like BERT on product names)



The problem with this algorithm is its inability to scale. It is also not able to learn the complex relationship between items.

Resource Link

Prod2Vec (Variation of Word2Vec)

Like Word2Vec, Prod2Vec can be used to learn embeddings for products. Given a transaction where three items A, B and C have been bought, the model learns embedding by trying to predict A and C given B. This architecture is called skip-gram.

The idea was that when we try to generate embeddings using this way, the products that are bought together are closer in the embedding space but products that are categorically similar are closer in the vector space. Since the recommendation was served using the closest item in the vector space, the recommendation was not good because we need to recommend items that pair well with the item in cart, not most similar items. (Example: For pizza, the recommendation were other kind of pizzas instead of drinks or other dishes that pairs well with pizza)

Resource Link

NextItNet [paper]

It is a Sequential recommendation algorithm, for predicting the next item a user might be interested in. In its architecture, it uses dilated convolutions and residual connections to capture long-term dependencies in the session.

Though this algorithm is built to capture complex sequential patterns in session, it does not consider the meta-data of item(price, item_name, item_ctg_desc) and order(weekday, hour) and suffers from the Cold-start problem.

SR-GNN [paper]

It is a session based sequential recommendation algorithm which leverages graph neural networks to capture both sequential and semantic information in sessions.

It works by constructing a graph representation of sessions, where each item in the session forms a node in the graph. The edges between these nodes represent the temporal order of item interactions. The model then applies graph neural networks to learn meaningful representations of the items and their relationships within the session graph.

Like NextItNet, it does not consider meta-data of items and orders and it also suffers from the Cold-start problem.

SASRecF [paper]

It is also a sequential recommendation model that uses self-attention mechanism in its architecture.It considers meta-data of items and orders for recommendation but suffers from the cold-start problem.

Hyperparameters

FP Growth

- Minimum Confidence: Minimum probability of item being purchased given certain item has been added to cart
- Minimum Support: Minimum percentage of occurence of current and next item in order in the dataset

NextItNet

- embedding_size: Embedding size of items.
- block_num : Number of Residual blocks

SR-GNN

- embedding_size: Embedding Size of items
- step: Number of layers in GNN

SASRecF

- hidden_size: Size of embedding to generate
- inner_size: Size of inner layer of feedforward layer of attention mechanism
- n_layers: Number of transformer layers in transformer encoder
- n_heads: Number of attention heads for multi-head attention layer

Evaluation Metrics

1. Preicsion@k: Measure the proportion of relevant items among the top-k recommended items.
2. Recall@k: Measures how many of the relevant items were actually retrieved and included in the top-k recommended items.
3. HitRate@k: Measures the success rate of a RecSys in generating recommendations that contain items of interest within top-k recommended items.
4. Item Coverage@k: Measures the percentage of distinct items recommended within the top-k recommended items.
5. nDCG@k: Measures the quality of the ranking of relevant items within the top-k recommendations.

Explain measurable metrics used to determine the performance of the model and why.

Model Training and Validation

Same data was used for all iterations with the same train-test-val ratio of 8:1:1.

Iteration 1

- Models experimented on
  - NextItNet
- Hyperparam info
  - epochs: 50
  - train_batch_size: 8192
  - eval_batch_size: 1024
  - learning_rate: 0.0001
  - learner: adam
  - embedding_size: 128
  - block_num: 7
  - MAX_ITEM_LIST_LENGTH: 10

- Models experimented on
  - SR-GNN
- Hyperparam info
  - epochs: 40
  - train_batch_size: 8192
  - eval_batch_size: 1024
  - learning_rate: 0.0001
  - learner: adam
  - embedding_size: 500
  - weight_decay: 0.01
  - stopping_step: 5
  - step: 1
  - MAX_ITEM_LIST_LENGTH: 10

- Evaluation metrics detail
  - recall@5 : 0.5622
  - recall@10 : 0.694
  - ndcg@5 : 0.305
  - ndcg@10 : 0.3533
  - hit@5 : 0.5622
  - hit@10 : 0.6954
  - precision@5 : 0.1124
  - precision@10 : 0.0695
  - itemcoverage@5 : 0.8476
  - itemcoverage@10 : 0.8998

Iteration 2

- Models experimented on
  - SasRecF
- Hyperparam info

        - epochs: 14
        - train_batch_size: 512
        - eval_batch_size: 128
        - learning_rate: 0.001
        - Weight_decay: 1e-4
        - learner: adam
        - hidden_size: 256
        - inner_size: 512
        - n_layers: 2
        - n_heads: 2
    - Metrics
        - recall@5 : 0.4311
        - recall@10 : 0.5806
        - mrr@5 : 0.2635
        - mrr@10 : 0.2833
        - ndcg@5 : 0.305
        - ndcg@10 : 0.3533
        - hit@5 : 0.4311
        - hit@10 : 0.5806
        - precision@5 : 0.0862
        - precision@10 : 0.0581
        - map@5 : 0.2635
        - map@10 : 0.2833
        - itemcoverage@5 : 0.1085
        - itemcoverage@10 : 0.1322

Final Model

After iterating and experimenting through all models, SR-GNN gives promising results.

Best Hyperparameters

- epochs: 50
- train_batch_size: 8192
- eval_batch_size: 1024
- learning_rate: 0.0001
- learner: adam
- embedding_size: 500
- weight_decay: 0.01
- stopping_step: 5
- step: 2
- MAX_ITEM_LIST_LENGTH: 10

Model Output

- recall@5 : 0.4856
- recall@10 : 0.629
- ndcg@5 : 0.305
- ndcg@10 : 0.3533
- hit@5 : 0.4856
- hit@10 : 0.629
- precision@5 : 0.0971
- precision@10 : 0.0629
- itemcoverage@5 : 0.8225
- itemcoverage@10 : 0.8852

Deployment

The model was deployed by exposing an API created by FastAPI which was dockerized and hosted in an EC2 instance.

Productionzation

Questions to address:

- How can the models be deployed?
- Where and how can the result be exposed?

Next Steps

- List down the tasks/experiments that can be performed to improve the model performance.
- Further analysis on the data

Tasks

- SASRecF, being able to capture item's contextual features, can be enhanced by using high level item features like embeddings from name.
- Graph based models can be used to generate embeddings which could be used in other models

Conclusion

The project was halted until further notice and there is still significant development that can be done.

Appendix

- List of initial Variables
- Hyperparameters grid
- Final model parameters
- Models
- Powerpoint Slides
- Miscellaneous Data References & Files
- Repository link