Ingredient-Based Recommendations: Technical Documentation

🌟Scope Summary

This documentation focuses on technical ideations, designs and implementations from data enablement for building an ingredient-based recommendation system, to implementing POCs for such recommendation systems using methods such as collaborative filtering and content-based filtering using Google Cloud Platform. Some technical evaluations are briefly discussed in this documentation, and proposed next steps in the context of the product roadmap are summarized at the end.

🛠Data Enablement

Based on our product design and technical ideations, we identified the following main data sources that will be used for building the recommendation system -

1. 2022 store sales data for Meet Fresh location in Plano, Texas (provided by Meet Fresh)
2. ingredient and product item attribute data (requires data collection efforts)
3. customer-level rating data (scaled 0-3) for each ingredient (requires external mock data)

I. Store Sales Data

The only data provided directly by our Meet Fresh partner is store sales data from 2022 for store location in Plano, Texas. Raw data was provided in pdf format, and some data extraction and cleaning efforts were required to convert data into tabular format and stored in BigQuery *dsxl-ai-advanced-program.meetfresh.ft_store_sales_plano_2022*

👉Store sales data can be leveraged to provide popular ingredient toppings and top seller products within a specified time window using simple sorting.

Quickly checking primary key presence -

group_id_present	article_id_present	cnt	rate
false	true	10	0.35752592062924...
true	true	2787	99.6424740793707...

~ 0.3575% group_id is missing in sales data table, but the main item id (article) is 100% present. Missing group_id poses no significant issue.

II. Ingredient, Product Item Attribute Data

Ingredient-level (**dim_ingredient**) and product-level (**dim_product**) item attribute data were collected from Meet Fresh USA site. The following steps were conducted for initial data processing -

1. Data enrichment was done for items missing descriptions using ChatGPT
2. Primary key mapping (ingredient_id, product_id) was done for dim_ingredient and dim_product tables, using values from store sales data

dim_ingredient table is stored in *dsxl-ai-advanced-program.meetfresh.dim_ingredient* and dim_product table in *dsxl-ai-advanced-program.meetfresh.dim_product*

Note that some items appearing in store sales data do not appear in dim_product/dim_ingredient tables and vice versa. High-level analysis was done to scope overlapping between two data sources to determine primary data scope for building POC.

1. Items in dim_product and dim_ingredient but not in sales data
Only looking at dim_product and dim_ingredient tables -

product_id_present	cnt	rate
true	117	91.40625
false	11	8.59375

ingredient_id_present	cnt	rate
true	21	87.5
false	3	12.5

(item presence in dimension tables)

11 out of 117 products (8.59%) in dim_product do not appear in sales data table, and 3 out of 21 ingredients (12.5%) in dim_ingredient do not appear in sales data table.

2. Items in sales data but not in dim_product and dim_ingredient
Within items that are present in sales data, only 33.41% of them can be identified in either dim_product or dim_ingredient tables -

sales_data_present	dim_data_present	cnt	rate
true	false	261	63.19612590799...
false	false	14	3.389830508474...
true	true	138	33.41404358353...

(item presence in sales table)

This means 63.19% items from sales table cannot be reliably mapped back to our dimension tables.

👉Ideally, every item from sales data table should exist in dimension tables. Due to lack of consistent data sources, we need to treat one dataset as primary data for defining product and ingredient item scope. Due to missing item attributes (i.e. description and image) for items from sales data table that are not in dimension tables, we use dim_product and dim_ingredient as our product and ingredient item scope.

III. Customer-Level Rating Data

In order to leverage collaborative filtering algorithms for our recommendation systems, customer-level rating data is needed. This type of data doesn't currently exist for Meet Fresh. Thus we need to utilize mock data that is adapted to Meet Fresh context.

Amazon Review Data

We first explored Amazon review data (2018) that comes with a smaller subset under the Grocery and Gourmet Food category. This subset dataset is stored in a GCS bucket for some initial exploration in Vertex AI.

The Amazon review dataset contains product id (mapped to **item_id**), reviewer id (mapped to **customer_id**), 1-5 scale rating (mapped to **rating**), and review time (mapped to **unix_review_time**). This is the minimum amount of information needed to build a collaborative-filtering based recommendation system.

	item_id	customer_id	rating	unix_review_time
0	1888861614	ALP49FBWT4I7V	5.0	1370304000
1	1888861614	A1KPIZOCLB9FZ8	4.0	1400803200
2	1888861614	A2W0FA06IYAYQE	4.0	1399593600
3	1888861614	A2PTZTCH2QUYBC	5.0	1397952000
4	1888861614	A2VNHGJ59N4Z90	4.0	1397606400
5	1888861614	ATQL0XOLZNHZ4	1.0	1392940800
6	1888861614	A94ZG5CWN70E7	5.0	1385856000
7	1888861614	A3QBT8YC3CZ7C9	3.0	1383696000
8	1888861614	AGKW3A1RB1YGO	5.0	1380672000
9	1888861614	A1B65IWLUVOUQB	5.0	1378425600

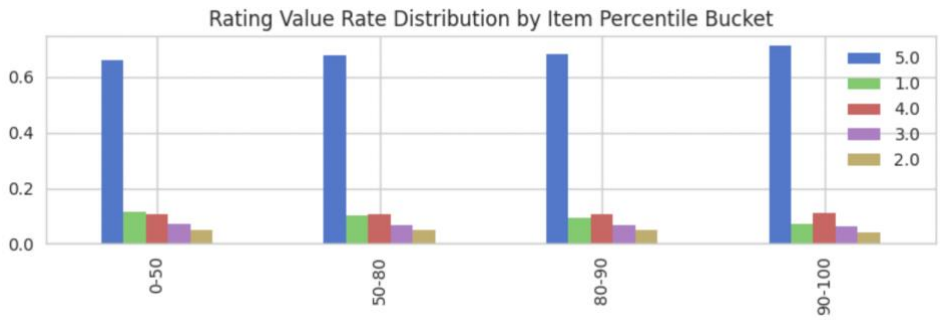
The following was done to scope and understand the original dataset in order to generate mock data for Meet Fresh -

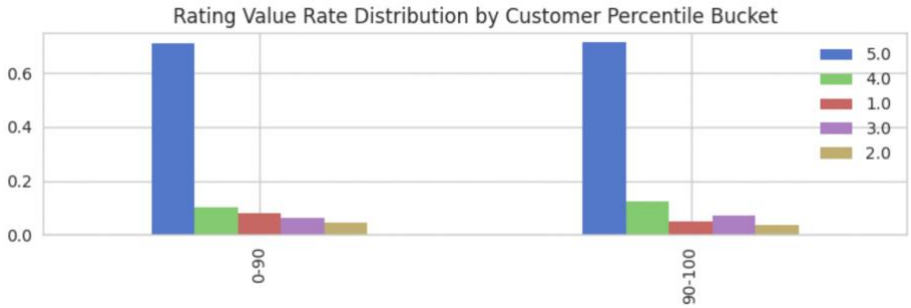
1. Both item and customer rating count distributions from Amazon review data are right-skewed, with lumps of review ratings concentrated at a small portion of top-reviewed items and super-reviewer customers.

```
amazon_df['item_id'].value_counts().describe()

count    283507.00000
mean         17.89783
std         107.50136
min           1.00000
25%           1.00000
50%           3.00000
75%           9.00000
max       11526.00000
Name: item_id, dtype: float64
```

```
amazon_df['customer_id'].value_counts().describe()

count    2.695974e+06
mean     1.882125e+00
std      3.187802e+00
min      1.000000e+00
25%      1.000000e+00
50%      1.000000e+00
75%      2.000000e+00
max      1.103000e+03
Name: customer_id, dtype: float64
```

2. Rating value distributions over different item and customer percentile buckets are as follows -





Top-reviewed items (90 - 100 percentile) have a slightly higher proportion of 5-star ratings and lower 1-star ratings. Super-reviewer customers give out significantly less 1-star ratings, and slightly more 5 and 4-star ratings.

We could proceed with conducting stratified sampling to assemble a dataset that makes sense for Meet Fresh ingredient rating context. However, Amazon rating data is not exactly the same as Meet Fresh product design, where ratings reflect customer preference before purchase.

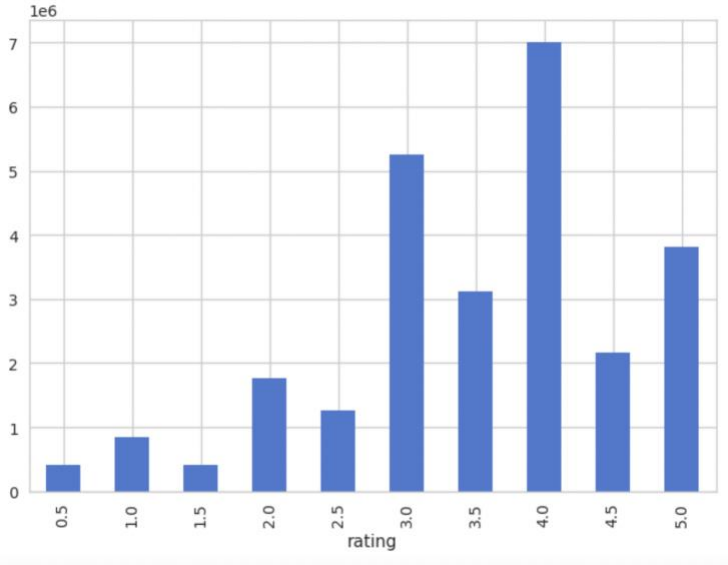
Movies Rating Data

Another mock data source we explored is the Movies dataset, from which we selected only the dataset containing user level rating for movies. We believe this dataset could be better suited to simulate preference ratings needed for Meet Fresh purposes. This dataset is uploaded to our GCS bucket.

Similar to the Amazon review dataset, the movies rating dataset contains movieId (mapped to **item_id**), userId (mapped to **customer_id**), 1-5 scale rating (mapped to **rating**), and rating timestamp.

	customer_id	item_id	rating	timestamp
0	1	110	1.0	1425941529
1	1	147	4.5	1425942435
2	1	858	5.0	1425941523
3	1	1221	5.0	1425941546
4	1	1246	5.0	1425941556

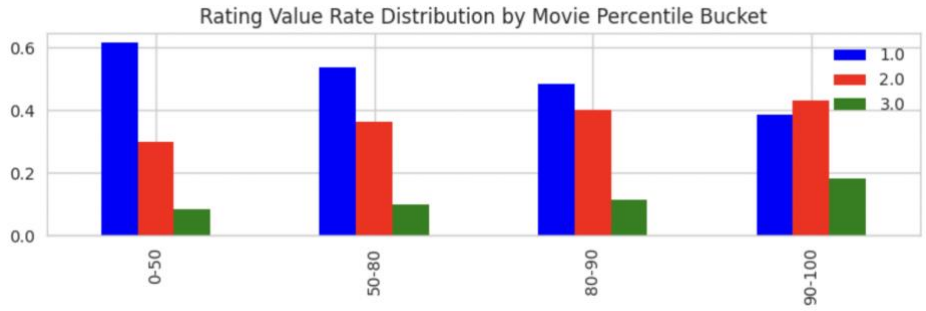
Different from Amazon review data, movies rating value takes on any value from (0.5,1,1.5,2,2.5,3,3.5,4,4.5,5). To simplify data exploration and align movies rating with Meet Fresh 0-3 scale rating, we use the following rating distribution and mapping rules -



1. Any unrated movies by a user in movies rating data corresponds to Meet Fresh rating 0 - customer has not provided information about their preference
2. Movies rating 3.0 - 3.5 corresponds to Meet Fresh rating 1 - customer has indicated mild preference (not negative sentiment as movies rating 0.5 - 2.5)
3. Movies rating 4.0 - 4.5 corresponds to Meet Fresh rating 2 - customer has indicated medium preference
4. Movies rating 5.0 corresponds to Meet Fresh rating 3 - customer has indicated strong preference

The following was done to scope and understand how this dataset could be adapted for Meet Fresh -

1. Ratings with 0.5 - 2.5 are excluded from Meet Fresh mock data, as Meet Fresh rating implies positive preference by design, but movie ratings could suggest negative sentiment. Therefore, we only keep movie ratings that at least indicate some positive sentiment to be better aligned with Meet Fresh design.
2. Similar to Amazon review data, both movie and user rating distributions are highly skewed, with lumps of movie ratings concentrated at a small portion of top-reviewed movies and super-reviewer users.
3. Rating value distributions over different movie and user percentile buckets are as follows -



The more reviews movies have, the higher proportions are for ratings 2 and 3 and lower proportion for rating 1, meaning popular movies get higher ratings. This could potentially align with Meet Fresh item rating distributions, where more popular ingredients such as herbal jelly, taro balls, get more and higher ratings.

The top reviewers in general give out lower ratings, implying seasoned movie reviewers are less liberal in providing high ratings. This may or may not reflect Meet Fresh customer ratings in reality.

Generate Mock Data

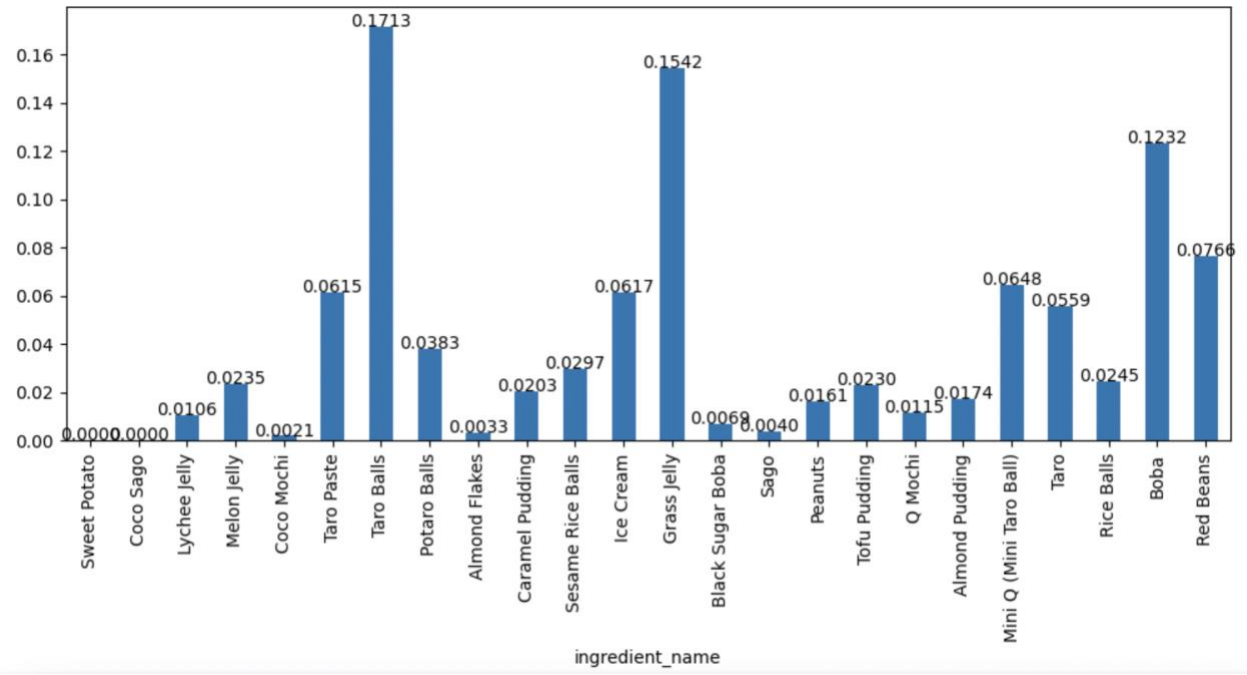
We use movies rating data source as basis for generating mock data. Steps are outlined in this section.

1. Ingredients Sales Quantity Distribution

In order to conduct stratified sampling for each of 24 ingredients defined in scope, we first need to obtain an approximate distribution for these 24 items. To estimate the true distribution of ingredients quantity distribution, we first use sales data from 2022 and aggregate sales quantity for these ingredients - including ingredients separately ordered as additional toppings, and ingredients already contained in sold products.

This was done in BigQuery - process can be found here and the final data table is stored in *dsxl-ai-advanced-program.meetfresh.ft_agg_ingredient_sales_quantity_distribution*

The distribution of ingredient sales quantity is as follows -



Sweet Potato and Coco Sago appear in dim_ingredient but do not appear in sales data as any additional topping or any product containing these two ingredients.

Sampling was done for each ingredient distribution and average value was taken to estimate the true distribution.

2. Stratified Sampling for Ingredients

The underlying assumption for this step is that popular ingredients, just like popular movies, get more ratings and especially higher ratings overall. Based on estimated ingredient distribution from the previous step, we conducted stratified sampling for each ingredient ranked from most popular to least popular, and only sampled from a certain item percentile. Vertex AI notebook with this process can be found here.

As last steps, Meet Fresh ingredient_id was mapped back to the mock dataset to replace original movie item_id, and sanity check was conducted to make sure customer rating distribution makes sense and match what we were expecting -

```
count    155594.000000
mean          2.640256
std           2.228229
min           1.000000
25%           1.000000
50%           2.000000
75%           3.000000
max          21.000000
```

It turns out that customers could rate at least 1 ingredient, and at most no more than 22 ingredients (excluding the 2 ingredient items that did not appear at all in sales data), and the overall distribution is skewed towards less than 5.

This mock data is stored in the same GCS bucket as a csv file. It is then loaded to BigQuery as a native table *dsxl-ai-advanced-program.meetfresh.ft_customer_ingredient_ratings*

😬Now we are ready to move on to prototype implementation stage.

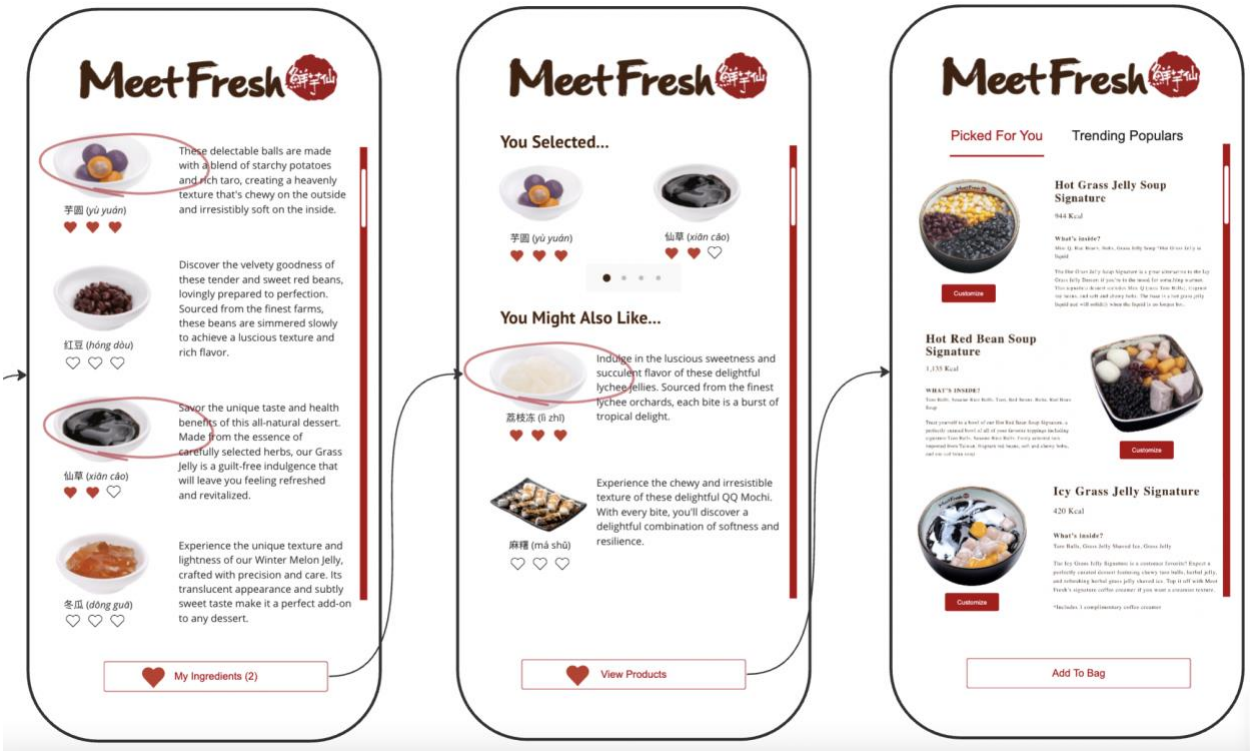
🤖Model Prototyping

We use the following sections to outline models prototyped for ingredient-level and product-level recommendations. Sections I and II describe ingredient-level recommendations using Collaborative Filtering and Content-Based Filtering approaches, and section III describes product-level recommendations using a simple Content-Based Filtering approach.

I. Collaborative Filtering Using Ingredient Ratings

Customers provide ratings for the ingredients shown to them upon entering the “Discover Ingredient” menu option. Meet Fresh ratings are on a 1-3 scale, with 1 being mild-level interest, 2 being medium-level interest and 3 being strong interest in a particular ingredient item. Customer-ingredient pair without a rating is the space where we will conduct collaborative filtering to discover recommendations.

Note that in this context we do not have a 0 rating, which usually means customers explicitly indicate that they are NOT interested in an ingredient item. Our Meet Fresh rating design is based on the assumption that customers do not provide negative feedback, as they know very little about the ingredient items and would like to discover more based on the few items that they know they like (this ties directly back to our customer needs identified in our need-finding phase). Therefore, we assume that all unrated items will be considered as potential recommendation candidates.



Collaborative filtering approach is detailed here. PySpark framework is used for this approach. In reality, ALS model process needs to be run in real-time with each customer's ingredient rating selections along with other customers' rating data to output predicted ratings for each ingredient for that customer. Ingredients based on sorted predicted ratings are provided in UI to customers.

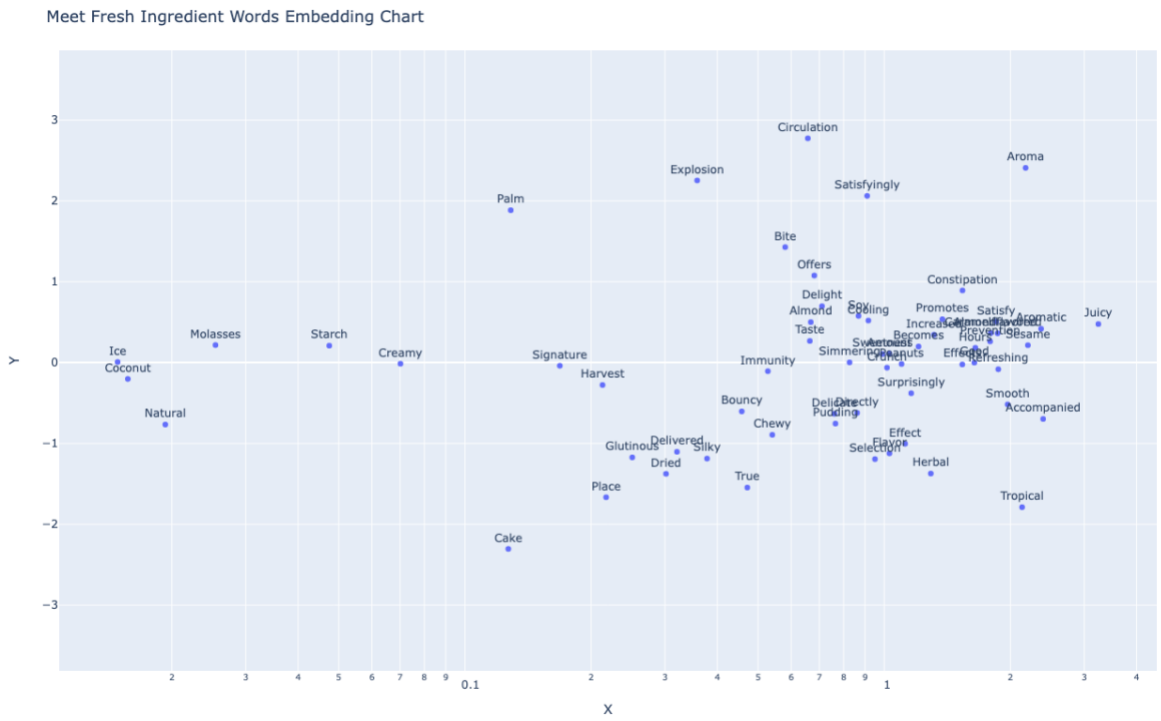
II. Content-Based Filtering Using Custom-Trained Meet Fresh Embedding

There are multiple ways to conduct content-based filtering, directly with item similarities or in conjunction with customer-ingredient rating matrix. The common step before content-based filtering is to obtain features that describe each ingredient. Again, there are multiple ways for this, from basic techniques such as TF-IDF using ingredient text descriptions, using pre-trained models to acquire embeddings, or custom training our own embedding for Meet Fresh ingredients.

1. Generate Custom Embedding

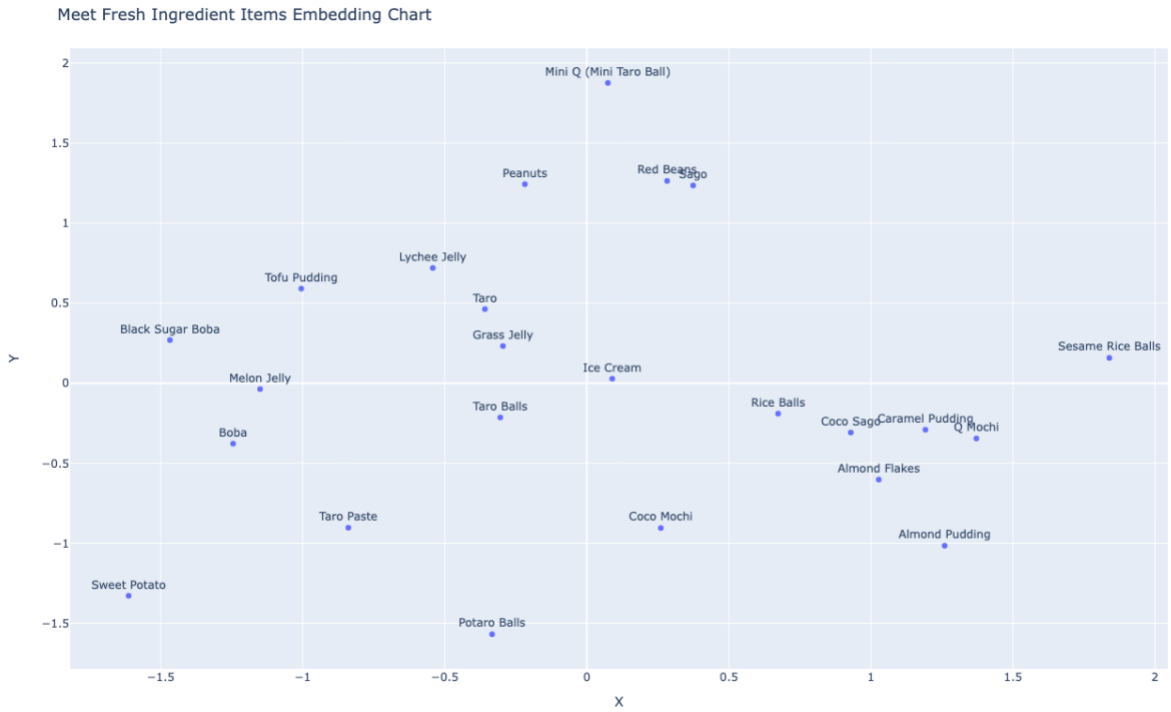
In our prototype, we explore custom training Meet Fresh embedding using ingredient text descriptions. A shallow neural network is used for creating the CBOW model that outputs weights that are used as embeddings. The full process is detailed here.

Using PCA and reducing embedding dimension to 2D, we plot the ingredient word embedding space as follows -



The outcome depends heavily on the semantic meanings given by ingredient text descriptions. We have limited text descriptions for Meet Fresh ingredients (and in some cases we utilized ChatGPT to conduct data enrichment for ingredients descriptions). We believe that with more comprehensive item descriptions, the embedding model is able to pick up more semantic content for these ingredients.

We used word embeddings to construct Meet Fresh ingredient embeddings visualized as follows -



2. Content-Based Filtering

Using ingredient embeddings as features, we utilized a weighted average rating approach by computing a customer feature matrix. This matrix is obtained by multiplying customer_ingredients rating matrix and ingredient_feats embedding matrix. Once customer feature matrix is calculated, it's convenient to generate recommendations by computing similarity measure between each customer feature vector and ingredient feature vector.

Masking the rated ingredients from our predicted ratings result (since it doesn't make sense to recommend already rated ingredients to customers), we obtain top 5 recommended ingredients to each customer as follows -

```
customer_id 1: ['Q Mochi', 'Melon Jelly', 'Rice Balls', 'Grass Jelly', 'Mini Q (Mini Taro Ball)']
customer_id 2: ['Q Mochi', 'Melon Jelly', 'Rice Balls', 'Grass Jelly', 'Tofu Pudding']
customer_id 4: ['Q Mochi', 'Mini Q (Mini Taro Ball)', 'Grass Jelly', 'Ice Cream', 'Tofu Pudding']
customer_id 5: ['Q Mochi', 'Mini Q (Mini Taro Ball)', 'Grass Jelly', 'Melon Jelly', 'Tofu Pudding']
customer_id 6: ['Rice Balls', 'Melon Jelly', 'Q Mochi', 'Almond Pudding', 'Coco Sago']
customer_id 7: ['Rice Balls', 'Melon Jelly', 'Grass Jelly', 'Tofu Pudding', 'Coco Sago']
customer_id 8: ['Boba', 'Black Sugar Boba', 'Mini Q (Mini Taro Ball)', 'Peanuts', 'Coco Mochi']
customer_id 9: ['Tofu Pudding', 'Coco Sago', 'Rice Balls', 'Black Sugar Boba', 'Almond Flakes']
customer_id 10: ['Grass Jelly', 'Tofu Pudding', 'Coco Sago', 'Peanuts', 'Rice Balls']
customer_id 11: ['Grass Jelly', 'Rice Balls', 'Peanuts', 'Coco Sago', 'Ice Cream']
```

III. Content-Based Filtering for Product-Level Recommendations

In this final step, we are able to use customer ingredient ratings data (updated and modified) and product attribute data to generate product-level recommendations to customers. The detailed process can be found here.

For product-level recommendations, we could simply use ingredient components to construct a feature space. We still need to utilize customer-ingredient rating data, but modified and updated after customers have completed final selection of ingredients, since ingredient-level recommendations are provided first based on initial ingredient ratings, and customers could select additional ingredients recommended to them to proceed to product recommendations.

In reality, we need to run product-level recommendation processes using the updated customer-ingredient data, and since we do not require customers to provide ratings for ingredients recommended to them, we need to modify the customer-ingredient interaction matrix by only using 1/0 to indicate whether an ingredient is selected up until when customer proceeds to product recommendation stage.

Extra cautious about the dimension of customer-ingredient matrix and product-ingredient matrix, as depending on data source, ingredient items do not always line up, and products without any detailed ingredient breakdowns are excluded from recommendations.

Finally, product recommendations are made by multiplying the customer-ingredient matrix and product-ingredient matrix, which essentially produces a weighted average of customer preference for products. No masking is needed here, unlike ingredient-level recommendations, as customers are never asked to provide ratings for products during this process, and therefore all top products within the recommendation space could be displayed to customers.