Substitution Techniques for Grocery Fulfillment and Assortment Optimization Using Product Graphs

Amit Pande, Aparupa Das Gupta, Kai Ni, Rahul Biswas, Sayon Majumdar
{amit.pande,aparupa.dasgupta,kai.ni2,rahul.biswas,sayon.majumdar}@target.com
AI Sciences, Target Corporation, USA

ABSTRACT

Identifying substitutable pairs or groups of products is key to building relevant product assortment for brick-and-mortar stores as well as to efficiently handle out-of-stock scenarios. In this work, we describe the unique challenges with a retailer's data to identify substitutable product pairs from a large catalog and nationwide store transactions. We apply some of the well established approaches in data mining and machine learning to customer store purchase data and product attributes data to generate networks of substitutable products. This paper presents a novel application of substitutable product networks integrated with an assortment optimization engine that was developed in-house to select the optimal assortment of products for the stores. The outcomes from a large scale experiment conducted across 120 stores within United States demonstrates a unit sales lift in excess of 11%. In another set of tests, we analyze the performance of various algorithms for product substitution and fulfillment when customers encounter out-of-stock scenario while shopping for groceries for same day delivery.

KEYWORDS

Product Substitutes, Data Mining, Graph, Assortment Optimization

ACM Reference Format:
Amit Pande, Aparupa Das Gupta, Kai Ni, Rahul Biswas, Sayon Majumdar. 2020. Substitution Techniques for Grocery Fulfillment and Assortment Optimization Using Product Graphs . In *Proceedings of 16th International Workshop on Mining and Learning with Graphs (MLG '20). August 22–27, 2020, San Diego, CA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Traditional brick-and-mortar stores continue to dominate the grocery shopping experience, but e-grocery is growing in popularity. According to a recent study, a quarter of American households are buying food online and total U.S. online grocery sales will rise to nearly $30 billion by 2021 [1]. To compete with tech-savvy retailers such as Amazon, stores are beefing up delivery options for customers, such as same-day grocery delivery, in-store pickup and drive-up services. These efforts are designed to improve the customer experience, which help to improve engagement and retention.

At the heart of all this is the strategic decision to focus on the idea of using stores as hubs. The traditional retailers realized that instead of closing all the stores and fighting behemoths like Amazon for online turf, it is much more convenient to use stores-as-a-hub and fulfill the orders from there. Fulfillment via stores is significantly cost-efficient than via the fulfillment centers. This is the key to improving digital performance of Krogers, Walmart and Target. Making stores the hub for fulfillment implies challenges for planning assortment for the customers as well as handling the scenarios when a customer is looking for an out-of-stock (OOS) product. OOS happens all the time in digital fulfillment, however, in the scenario of store-based fulfillment, the frequency of OOS increases. In the case of online purchases, the order can be fulfilled by any of the distribution center. However, for same-day fulfillment via stores in the form of store-pickup or drive-up or home delivery, the inventory is transient. While the digital customer is placing an order for mangoes, another customer might have bought the entire crate of mangoes from the stores and it is not ticketed yet. Or a reportedly fresh product might be found to be stale when the shopper goes to pick it up. The overall inventory level in stores is lower than fulfillment centers. In all these scenarios, recommending substitutable products to the customers are of paramount importance.

Product recommendation is a well-studied part of customer personalization experience in digital retail for decades. The old Collaborative Filtering or k-Nearest Neighbor based approaches can be naively used for product substitution recommendations with some success. Content based approaches [2, 4, 11] use semantic content such as product description, attribute data and reviews to define similar and substitutable products. Collaborative filtering [9, 13] is a context based approach which selects information based on the interest and opinion of other users. Item-to-item collaborative filtering matches each of the user's purchased and rated products to similar products, then combines those similar products to generate recommendations. Since customers view similar products before shopping the product they like, co-viewed products form the crux to recommend similar or substitutable products to the customer. However, store sales data lack knowledge of other products customer looked at in the aisles and considered before making the final purchase. In the digital domain, newer algorithms [7, 16] use visual cues to recommend similar products to customers. Other approaches [10] combine both content based and co-purchase based signals from online interaction of customers.

Stores, unlike digital, have limited shelf space and a retailer has to intelligently select its assortment to satisfy the local demands. Determining the best product mix and inventory size are of strategic importance to a retailer looking to cash in on store fulfillment.

MLG '20, August 22–27, 2020, San Diego, CA

ACM ISBN xxx-x-xxxx-XXXX-X/xx/xx…$15.00
https://doi.org/10.1145/1122445.1122456

Assortment planning refers to the problem of selecting the right set of products to offer to a group of customers to maximize the revenue that is realized when customers make purchases according to their preferences. It is a trade-off between the shelf space and the products we want to show to the customers locally. Substitutability is a key factor in deciding the right cohort of products within a category. Assortment planning is mostly seen as an optimization problem [5]. Some earlier works like by Kök et al. [8] use the multinomial logit demand model to determine substitutes. None of the past work leverages graphs and machine learning techniques for product substitutions and its impact on assortment planning.

To the best of our knowledge, this is one of the few works leveraging machine learning techniques for product substitutions in assortment optimization formulations as well as same-day delivery substitutions. The main contributions of this work are as follows:

(1) This work discusses the data challenges as well as related performance of context based and content based recommendations for product substitutes based on product descriptions and store purchases.
(2) A novel technique to build product substitution graphs using customer store trips has been introduced.
(3) A Graph Convolutional Network (GraphSWAG) with unsupervised loss is used to integrate two data sources (product description and product co-purchase behavior) and generate improved substitutions. Embeddings based on product attribute descriptions were also generated as a baseline.
(4) Performance evaluation of algorithms is done for online grocery delivery OOS scenarios. GraphSWAG outperforms other product recommendation techniques in different substitution scenarios and scales well.
(5) The substitutable product networks were integrated to a store assortment planning exercise done over 120 stores for cookies and crackers assortment. Lift obtained in units sold and total transactions was in excess of 11% .

The rest of the paper is organized as follows: Section II gives an insight on data and challenges in processing it. Sections III and IV give details of product substitution techniques followed by their evaluation for grocery OOS scenarios in digital fulfillment. Section V gives a short summary of Assortment Optimization experiments and key results. Section VI concludes the work and gives direction for future work.

2 DATA CHALLENGES AND MOTIVATION

Traditional brick-and-mortar retailers derive a major chunk of their revenue from store sales. This translates to billions of transactions every year by customers. Of the entire population of customers, a significant portion still use cash, checks or other non-identifiable payments. Only a subset of customers enroll in loyalty program or add their cards to their wallets and are uniquely identifiable. For this work, we needed to leverage the data available from the uniquely identifiable customer base. This limited the size of the population selected for identifying substitutable products.

In order to generate networks of substitutable products, we partitioned the product catalog by categories. For instance, within grocery and essentials, there are about 300 different categories such as crackers, cookies, milk and yogurt. This partitioning step is very helpful because it limits the scope of substitution. The likelihood of substitution from cookies to milk is almost zero. Hence, we can build sub networks for each category. This facilitates computation of networks in parallel across several categories at the same time. Due to product differentiation and varied offerings across different geographical locations, each of these categories comprised of several products.

Table 1: Customer Cookie Purchase Across Store Trips

CustomerID	Transaction Date	Product Description	Product ID
6034	2019-01-16	Oreo Double Stuf Cool Mint 15.25oz	123
6034	2019-01-20	Oreo Double Stuf Cool Mint 15.25oz	123
6034	2019-02-02	Oreo Double Stuf Cool Mint 15.25oz	123
6034	2019-02-19	Oreo Double Stuf Cool Mint 15.25oz	123
6034	2019-02-19	Oreo Original 14.3oz	234
6034	2019-02-28	Oreo Double Stuf Cool Mint 15.25oz	123
6034	2019-03-29	Oreo Thins Mint 10.1oz	345
6034	2019-03-29	Oreo Original 14.3oz	234
6034	2019-06-19	Oreo Thins Mint 10.1oz	345

It is difficult to infer substitution data from a customer's store trip behavior. In online world, it is easier to infer substitution based on similar products a customer viewed or added to cart before purchasing a product. This has been exploited a lot to build a sequence of user product interactions which is used for context based recommendation algorithms. Accurately tracking a customer's interactions with products using cameras may invoke privacy concerns from customers. Thus, this approach was dropped and we took a more realistic approach to look at the customer's behavior. A significant portion of store transactions is done by identified customers (identified through loyalty program or usage of credit card transactions). We looked at store purchases of identified customers and built offline sessions. We used transactions of over 100 million customers for the past one year to create offline sessions. For instance, we created a session for a given customer that comprised of all past purchases for last one year. Such sessions were created for each category. In Table 1, we show a snapshot of a customer's purchases from the cookie category in 2019. We observe that across the four trips that he made to the stores he bought Oreo Double Stuf Cool Mint, Oreo Original , Oreo Thins Mint. So the purchase session for this customer within cookies category for this snapshot of data would comprise the IDs [123, 234, 345].

In Figure 1 we show three subnetworks of the complete graph of substitutable cookies, (i) traypacks/ multi-packs/ variety packs/ munch packs , (ii) TimTams, and (iii) Just Cookies. The length of the edges is proportional to the Association Score, that we describe in the next section, shared by the two products. Here, we observe that the three clusters of substitutable products do not share any edges indicating they are not to be considered as substitutable.

The structured attribute data for products in the catalog comprise of information regarding brand, flavor, health and wellness related attributes and other product features including product titles that were provided by the vendors in the catalog. Instead of behavioral data, it is tempting to just use product-attribute dataset (which also has full coverage over range of groceries) and use them to recommend substitutable products. However, using pure attribute data may not lead to accurate results. Without going into the mathematics (which we shall discuss in later sections), we show some cookies and their substitutable counterparts, identified using store sessions

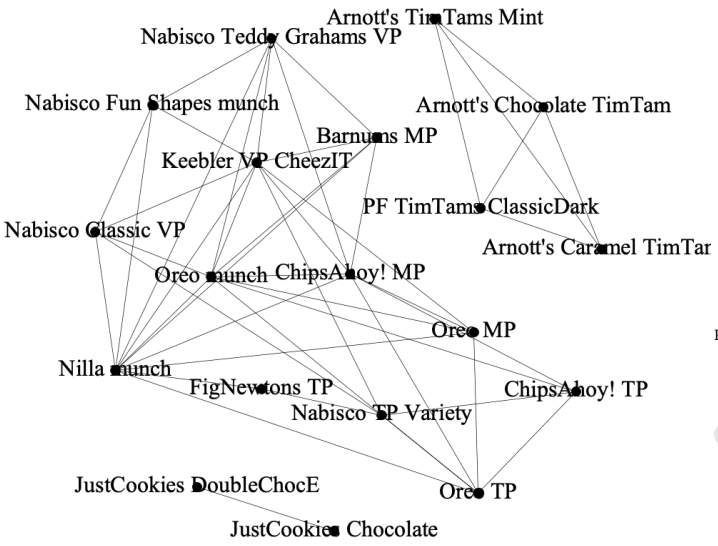


Figure 1: Substitutable product subnetworks for cookies category. Smaller distance indicate highly substitutable products.

Figure 2: Left - Original Cookies, Middle and Right - Cookies with high association score from purchase sessions. Middle column cookies have high score from attribute based algorithms too but Right column cookies have low score from attribute based approaches

in Figure 2. Pepperidge Farm's Pirouette Creme Filled Wafers differ slightly in flavors (Chocolate Fudge versus Chocolate hazelnut). Using store-sale signals, we also infer that Archer Farm's Creme-Filled Rolled Wafer are highly substitutable but attribute based methods fail to capture this nuance. Similarly, some sugar cookies were found to be substitutable with Valentine's day special cookies although product description doesn't mention the occasion/name 'Valentine'.

Apart from the catalog and transaction data other sources of data like product reviews were very sparse.

3 ALGORITHMS TO IDENTIFY SUBSTITUTABLE PRODUCTS

We leveraged purchase sessions to create the network of substitutable products for every category by evaluating the metric association score (AS) defined by the equation

$$AS(A \rightarrow B) = \frac{|A \cap B|N}{|A||B|} \tag{1}$$

where $|A \cap B|$ is the total number of customers who buy both product A and product B and $|X|$ is the total number of customers who buy only product X. This metric is also called the lift as defined by [15] and is commonly used for market basket analysis. Let N be the total number of customers in the transaction dataset. Dividing both numerator and denominator by N, we get

$$AS(A \rightarrow B) = \frac{P(A \cap B)}{P(A)P(B)} \tag{2}$$

We can infer that two products A and B are substitutable if $AS(A \rightarrow B) > 1$ since that implies $P(A|B) > P(A)$ by applying conditional probability given by equation 3 as

$$P(A \cap B) = P(A|B)P(B) = P(B|A)P(A) \tag{3}$$

In other words if the customer has bought product B then that increases the chance of buying product A. Note that the definition of basket in our context is the set of all products purchased by a customer from a single category over last one year. This makes the approach applicable for identifying substitutable products. Other approaches like mining negative associations [14] could also potentially be used for identifying substitutes. For our use case we keep the minimum support and confidence values as defined in [15] at zero.

Behavioral based approaches using browsing and co-purchasing data have been used in the past for identifying substitutable products. For the given retailer the customer traffic in store is significantly larger than the traffic on the website especially for certain categories like grocery and essentials. Hence, we applied Item-to-Item Collaborative Filtering [9] on store purchase behavior of customers for those categories. We created sessions out of consecutive store visits of customers in which they bought any product from a given category. The association score metric was used for item-to-item Collaborative Filtering to obtain pairs of products that could be deemed as substitutable. So if a given product is out of stock then we recommend the product with the highest association score metric to the customer as a substitute, provided that product is available in that store. We also tried other metrics like Jaccard similarity coefficient $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$ and the results were similar.

The primary drawback with purchase behavior based graph is the low coverage of products in the graph. In order to generate graphs with full coverage we leveraged the product attribute data

from the catalog and applied techniques from machine learning and data mining. We describe them in the following sections.

3.1 Simhash for Substitutes

Locality sensitive hashing (LSH) is a well established technique that hashes similar products into the same bucket with high probability. We implemented Simhash [4], one of the fastest LSH algorithms. Corresponding to each product, we have data which comprises the title of the product and a list of key-value pairs where the attribute name is the key and there is a value associated with that attribute name. We represented a shingle by each word in the title of the product and each of the key-value pair so each product was represented by a set of shingles. We hashed each shingle by MD5 hash function then used Simhash to reduce the dimension of the representation of products to a 64-bit hash string. The edge weights were evaluated as the inverse of the hamming distance between the hash strings of a pair of products.

3.2 Word2vec

Word2Vec is a group of models used for efficiently learning a standalone word embedding from a text corpus [11]. The product title and product attributes are considered as a sentence. Each attribute name and value is considered as a word. Next, a word2vec model (CBOW), the window size and other parameters are chosen using empirical evidence in performance metrics. This gives a unique representation to each attribute of a product. Once we obtain these embeddings, we take the mean of the embeddings of each attribute that is present to obtain the embedding for the product. This is called as word2vec model (w2v) in subsequent sections. The similarity between generated embeddings are then used to form networks of substitutable products.

Human Judgement and Attribute Weights

Based on empirical experiments, we observed that all attributes did not contribute equally to the notion of substitutability. In fact the key attribute for substitutability would vary based on the category in consideration. For instance, among razors, the retailer carries products from the same brand but targeting different genders so one has to ensure that if the customer was looking for Venus razors, they are not recommended Gillette for men since both the products share the same brand name. Hence, the attribute gender is very important for this category. However, for another category such as feminine products, this attribute is implicit. In order to capture the impact of such key attributes, we manually identified and created weights for those attributes with the help of business product owners and human judgement tools. The weights were used to generate recommendations using Simhash algorithm (simhash) and to build weighted word2vec models (w-w2v). For simhash, the hash string was created using the weights so that two items that share the same key attribute value are closer to each other after dimension reduction. We performed a weighted average of word2vec embeddings of the attributes to obtain product embedding in the later case.

We also tried other NLP based approaches like tf-idf but it ignored the importance of common key attributes like gender so the solution was less accurate for recommending substitutes. Also other similarity measures like Jaccard Similarity for sets of attributes or Cosine Similarity for vector representation of product attributes are computationally more expensive than other techniques.

3.3 GraphSWAG

GraphSWAG is a Graph Convolutional Network (GCN) suitable for weighted graphs [12]. GCNs are generalization of Convolutional Neural Networks for graphs. SWAG is capable of learning embeddings for nodes in web-scale graphs and deployed to generate recommendations for millions of product recommendations at Target.

Sampling is an important first step in Graph Convolutional Networks. As opposed to computer vision, where convolutional neural networks can use pixel proximity as a feature, GCNs do propagation guided by the graph structure. Therefore, for any given node, we need to efficiently select its neighbors for convolution. In the algorithm, the neighbor function, $N(v)$ denotes sampled set of neighbors for any given node $v \in \mathcal{V}$. In our use case of product substitutes, the larger the weight of the edge, higher the chances that the corresponding neighbor should be selected in sampling. After sampling, the selected neighbors need to be aggregated to their corresponding nodes for information clustering. The aggregation step is similar to convolution over nearby pixels in images and has the goal of aggregating information from neighboring nodes. However, a node's neighbors have no particular or natural ordering in graphs. Aggregators are used to aggregate all the neighboring nodes at the same distance. The *mean* aggregator, for example, would take an element-wise weighted mean of vectors in $N(v)$. We start with a random input embedding (or embedding generated from word2vec like models on product descriptions) for a node v. Let us call it h_v^0. A neighbor vector can be generated for each node as following:

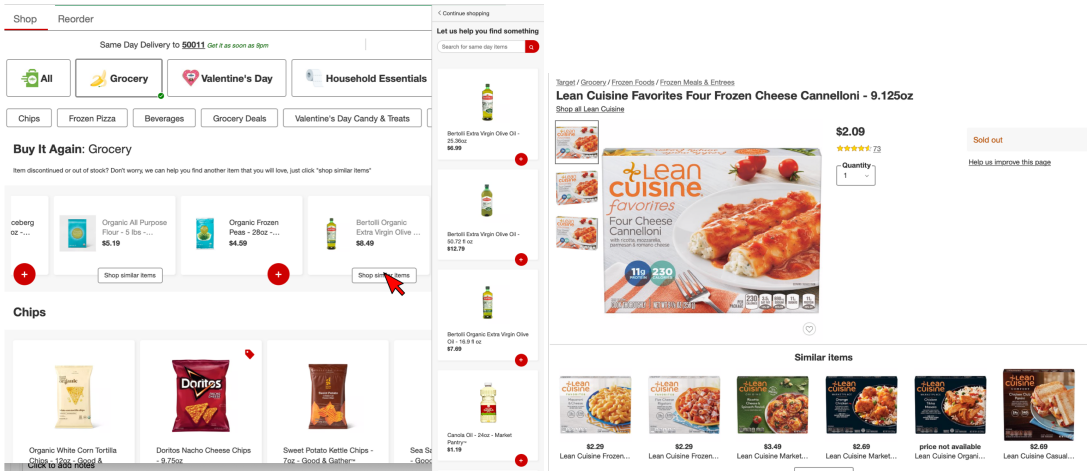
$$\mathrm{h}^k_{\mathcal{N}(v)} \leftarrow \pi_k(\{s(u,v)^\gamma \mathrm{h}^{k-1}_u, \forall u \in \mathcal{N}(v)\}) \tag{4}$$

Here $\mathrm{h}^k_{\mathcal{N}(v)}$ is the neighbor vector obtained for node v. π is the aggregator and $s(u,v)$ denotes edge weight between nodes u and v, determined using product substitution graphs discussed earlier and γ is a scalar. The node embedding for k-th iteration is given by:

$$\mathrm{h}^k_v \leftarrow \sigma(\mathbb{W}^k \cdot \mathrm{CONCAT}(\mathrm{h}^{k-1}_v, \mathrm{h}^k_{\mathcal{N}(v)})) \tag{5}$$

Here σ is the softmax function and $\mathbb{W}$ is a trainable model parameter. The model parameters can be learned using standard stochastic gradient descent and back-propagation techniques. We apply a graph-based loss function to the output representations, $z_u, \forall u \in \mathcal{V}$, and train the parameters $\mathbb{W}^k$ of the aggregator functions for $k \in \{1, \ldots, K\}$ via stochastic gradient descent. The graph-based loss function encourages nearby nodes to have similar representations, while enforcing that the representations of disparate nodes are highly distinct. Cosine similarity on the generated embeddings gives us the list of recommended substitutable products.

We trained two SWAG models, one based on online customer transactions (SWAG(o)) and considering all the grocery products as a single graph. However, in the second flavor, we trained SWAG based on substitutions-subgraphs based on store purchase graphs described earlier. In tuning the second flavor (SWAG(s)), we also observed that the neighbors required for aggregation must be small (3-10) instead of the larger number we considered in online graphs[12]. All the subgraphs were fed to the model as a single graph input,



(a) OOS substitutions on Same Day Grocery Delivery page

(b) Substitutions on Product Display Page

Figure 3: Customer is recommended similar/substitutable products when the desired product is locally out of stock.

along with word embeddings which serve as node embeddings for the graph model. Hyperparameter tuning was done using Skopt [6] and the objective was set to improve the click rate for the dataset.

4 DIGITAL GROCERY FULFILLMENT

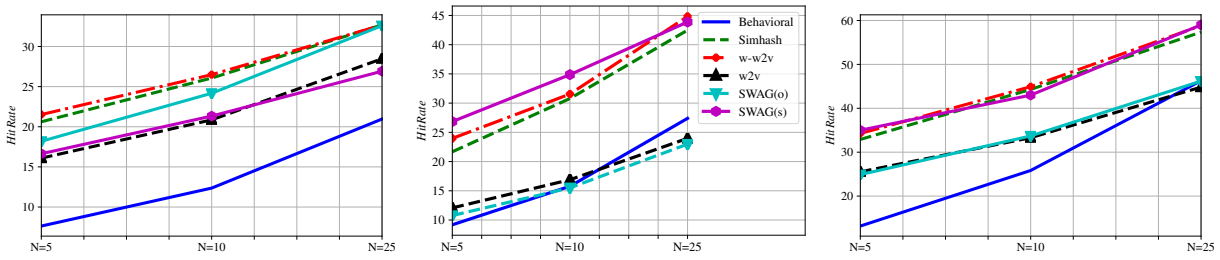
Figure 3 shows some snapshots of a customer browsing the website for grocery delivery and encountering OOS scenarios. In such scenarios, it is imperative that a retailer recommends customers substitutable products. We use such scenarios to evaluate the performance of the algorithms discussed in the previous sections.

We generated three different datasets for evaluation. Dataset 1 (Sessions) involved data from one month of customer grocery browsing sessions on the website. This dataset was huge but not exactly indicative of substitutions. We took the first product in session as seed product and considered other products in browse sessions as substitutes. However, it is possible that the customer was exploring different products (not similar) before he made a final choice. The main benefit of this dataset was its sheer size in order of tens of millions. The dataset had around 37M customer interactions. Dataset 2 (OOS) considers the scenarios as above where the product desired by the customer online is actually out of stock and the website recommends substitutable products. This dataset was smaller than dataset 1 and snapshots are shown above for such scenarios in Figure 3. The dataset had around 33M customer interactions. Dataset 3 (offline) involved the real-world scenarios where customers ordered something using same-day delivery service and it was found to be OOS after order was placed. A substitute was chosen by the customer using the App or employee interaction. This data was relatively new, small and highly personalized. We were able to access data across 1434 brick and mortar stores and it had 30K such transactions. The data contained the product id that was requested by the customer and the alternate product id that was selected in its place due to the requested product being out of stock in store. We report the offline evaluation of the approaches discussed in the previous section. Similar trends were obtained in online evaluation.

4.1 Measurements

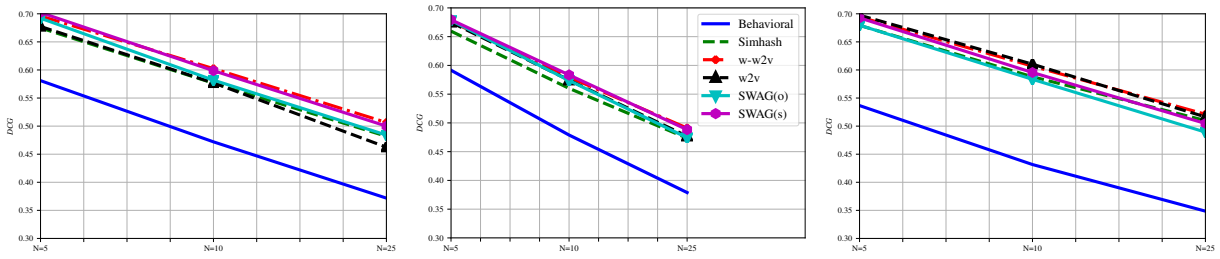
HitRate@topN is defined as the percentage of times, one of the top-N recommendation was selected by the customer. Figure 4 shows the performance of substitution algorithms for various values of N. Although a customer usually sees only top $N \sim 5-10$ recommendations, location specific unavailability of assortment (particularly in groceries) implies that a large number of recommendations may not be available in the stores at the time of purchase and hence not filtered out to the guests. The results are shown in Figure 4. DCG or Discounted Cumulative Gain is calculated with all products being given equal relevance (=1). When there is a hit at rank i, we increase the gain by $\frac{\log(2)}{\log(2+i)}$. The average values are reported in Figure 5. The following key observations can be made:

(1) Store session based Behavioral algorithm leveraging item-item collaborative filtering fared poorly in all the scenarios.
(2) Weighted word2vec algorithm fared better than naive w2v algorithm. However, the generation of weights required human feedback, which limits its scalability compared to others.
(3) SimHash performed exceptionally well for the three datasets. However, computing pairwise hamming distance is more expensive than nearest neighbor for embeddings. Like w-w2v, it also requires human feedback.
(4) GraphSWAG(s) trained on store data performed poor on sessions dataset but outperformed GraphSWAG(o) on real substitution tasks in OOS and offline dataset.
(5) SWAG(o) performed well for sessions dataset since it considered similar data for training. Sessions dataset has false positives due to customer browse behavior which was well captured by SWAG(o) but not by SWAG(s).
(6) SWAG(s) outperformed other algorithms for OOS and offline datasets. It was able to efficiently boost the performance over



(a) Sessions dataset (b) OOS dataset (c) Offline dataset

Figure 4: Hitrates of different algorithms for real-world datasets. w2v indicates word2vec, w-w2v implies weighted word2vec, SWAG(o) refers to SWAG trained on graph obtained from online browse sessions while SWAG(s) refers to GraphSWAG trained on store sessions described earlier.



(a) Sessions dataset (b) OOS dataset (c) Offline dataset

Figure 5: Discounted cumulative gain of different algorithms for real-world datasets. w2v indicates word2vec, w-w2v implies weighted word2vec, SWAG(o) refers to SWAG trained on graph obtained from online browse sessions while SWAG(s) refers to GraphSWAG trained on store sessions described earlier.

both behavioral and attribute based methods by combining the two in an unsupervised setting.

(7) Figure 5 shows that SWAG as well as w-w2v ranked relevant recommendations slightly higher than others, and did significantly better than behavioral graph based recommendations.

5 APPLICATION OF ITEM SUBSTITUTES FOR ASSORTMENT OPTIMIZATION

The Assortment Optimization model aims to maximize the revenue from sales by selecting the right subset of products from the catalog to display on the store shelves. Figure6 shows the components of the engine. The core of the engine is an optimization model that generates assortment recommendation for a given category and a given store by taking the following inputs: 1. **Product Sets**: Due to supply chain constraints, all products in the catalog are not available at all store locations. Every store has its own product set that is treated as its universal set and a subset of products is recommended by the assortment optimization model, 2. **Sales forecast**: The sales forecast of all the products in product set for the planning horizon, 3. **Product stickiness**: Product stickiness is a measure of customer loyalty that a particular product drives for the store, 4. **Business rules**: These are constraints like retaining products that are deemed mandatory or belong to self owned brands. These business rules translate to constraints in the optimization model, 5. **Product substitutes**: When a product is absent from the inventory, customers may purchase alternate products instead of this particular product, and hence the demand is transferred to substitutes.

5.1 Assortment Optimization Model

In this paper, we focus on two categories from the grocery division, cookies and crackers to describe the results from the model and the store tests that were conducted. In the model, we used behavioral product substitution network graph instead of earlier discussed techniques such as weighted word2vec or GraphSWAG model as an input to the optimization problem to represent product substitution. This was preferred for the following practical reasons:

(1) Although behavioral model performed poorly than weighted word2vec and GraphSWAG model in online substitutions, it is entirely possible that the behavior of store customers is entirely different than the online customers. The store customers see a curated assortment and hence their behavior and expectations are vastly different than online customers.

(2) Since behavioral graph was locally trained on store's customers' historical purchase behavior, it was more reliable to translate its scores as probabilities for the AO project.

Top Sellers Selected	Fair Sellers Dropped	Low Sellers Selected
#1, #2, #3 ... #162, #164, #165, #166, #167, #168, #170, #171, #172, #173, #175, #177, #178, #181, #182, #184, #185	#163, #169, #174, #176, #179, #180, #183, #186	#187, #188, #189, #190, #191, #194, #195, #204

Table 2: Model recommendations for cookie assortment in a sample store. Cookies are numbered by sales volume. It can be seen that the model picks up a few low selling cookies for the assortment to allow substitutability and demand transfer.

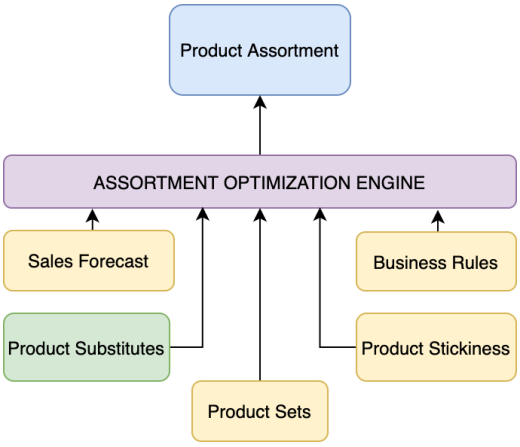


Figure 6: Simplified Assortment Optimization Engine Architecture

(3) In the first iteration of AO involving huge capital in the form of store A/B tests, we were biased to use explainable models instead of using a black-box deep learning model whose scores cannot be easily explained to other stake holders (site merchants, store leaders). Inspired by the success of the experiment, we plan to implement deep-learning models in subsequent tests.

Let N denote the number of unique products that a given store shelf can carry and x_i be a binary variable denoting whether product i is selected in the assortment. Let w_{ij} be the edge weight between products i and j in the substitution product network for this category. Let p_{ij} denote the proportion of customers who could potentially substitute product i with product j if product i is out-of-stock. This is given by $p_{ij} = w_{ij}/\sum_j(w_{ij})$.

Note that $w_{ij} < 1$ implies $P(i|j) < P(i)$, i.e., if the customer bought product j then that reduces the chance of that customer buying product i. Hence, we assume those pairs of products are non substitutable (i.e. $p_{ij} = 0$). We denote by s_i the forecasted sales in dollars for product i in the projected time frame for which the assortment planning is being executed.

We formulate the Assortment Optimization model as follows:

$$\max_i \sum_{i=1}^{N} x_i s_i + \sum_{i=1}^{N}\sum_{j=1}^{N} s_i p_{ij} x_j (1 - x_i) \tag{6}$$

subject to the constraints

$$\text{MinProductCount} \leq \sum_{i=1}^{N} x_i \leq \text{MaxProductCount} \tag{7}$$

$$x_i \geq \text{loyalty}_i - \text{loyalty}_{\text{threshold}} \tag{8}$$

Equation 6 represents the objective function for maximizing revenue. If a certain product i is not available, i.e. $x_i = 0$, the objective function captures the fraction of its sales which could potentially transfer to other substitutable products within the category. Equation 7 represents the business constraints on the maximum and minimum number of products to be selected in the assortment respectively. Constraint 8 ensures that if a product exceeds the threshold on the loyalty/stickiness metric then it is included in the assortment. Similarly, all other business rules can also be added as constraints in the optimization problem. The above formulation is an integer programming model that was implemented in GAMS and solved by CPLEX which is a large scale commercial solver [3].

Table 2 presents the results after running this model on cookies category for a single store with shelf space for 186 unique cookies. Due to supply chain constraints only 408 cookies from a total of over 2000 cookies were available from that category for this store. We removed all other constraints apart from maximum and minimum product count constraints. As shown in Table 2, the final recommendation generated by the assortment model included most of the top selling products in the list. Only 8 products from low sales ranked products were dropped and replaced by 8 other products which were ranked even lower in forecasted sales but could potentially absorb the demand from all other products that were not being selected.

The forecasted sales were generated from models that did not capture the interactions of demand across products that were on display for sale at the same time. Hence, when the projected sales forecast are large in value they overshadow the impact of substitutability. Only for products with low sales, we perceive an impact due to substitutability.

5.2 Hypothesis Testing in Stores

In the digital space the sheer high and tracked volume of visitors is sufficient to perform any A/B test with enough confidence on a given primary metric. In case of brick and mortar stores, it is difficult to assess the total number of customers visiting a given store. It is therefore impossible to compute certain primary metrics such as conversion as in digital counterpart. The stores are further categorized into markets, groups and regions. First and foremost, sensitivity analysis is done to understand the impact of sample size, N_s, on minimum detectable difference. In case of physical brick and mortar stores, the N_s is approximated as the product of total number of stores (where the actual test takes place) N_{STORES} and duration of test in weeks N_{WEEKS}. The minimum detectable difference Δ on the primary metric is inversely proportional to the N_s given as follows: $N_s = 2\left(z_{1-\alpha} + z_{1-\beta}\right)^2 \frac{\sigma^2}{\Delta^2}$,where z is the standard z-score, σ is the population standard deviation. We can further determine the minimum detectable difference for a range

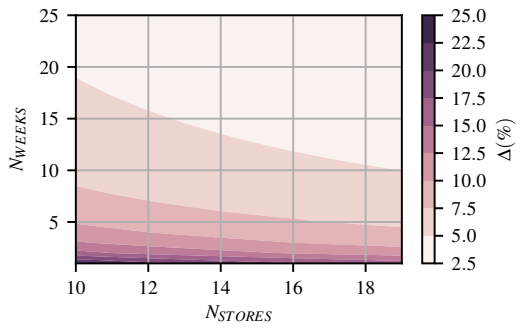


Figure 7: Contour plots of minimum detectable difference as a function of N_{STORES} and N_{WEEKS}

Metric	Market	Increment	Bounds	Significance
units	Atlanta	>9%	(-3, 22)	FLAT
trans	Atlanta	>9%	(-1, 24)	FLAT
units	Chicago	>12%	(2, 25)	YES
trans	Chicago	>12%	(2, 25)	YES
units	Houston	>15%	(5, 28)	YES
trans	Houston	>15%	(5, 27)	YES
units	Washington D.C	>4%	(-2, 15)	FLAT
trans	Washington D.C	>4%	(-2, 15)	FLAT

Table 3: Results of A/B tests in 120 stores at 95% confidence interval.

of N_{STORES} and N_{WEEKS}. Figure 7 shows contours of minimum detectable difference. The type I (α) and type II (β) error rates are set to 0.05 and 0.2 respectively.

In practice σ is computed from the stores sales and N_{WEEKS} is decided based on operating constraints. For this paper, experiments were designed across four different markets namely, *Atlanta*, *Chicago*, *Houston* and *Washington D.C*. In total 120 stores were selected for control and test group across 4 markets with each group containing 15 stores. Units sold (*units*) and number of transactions made (*trans*) were the primary metrics chosen for this experiment. The actual experiment was conducted for around 6 months, which then translates into $N_s = 390$ samples. The minimum detectable difference computed for this experiment was around 2.5%. Table 3 summarizes the results of the A/B test performed across 4 different markets. The percentage increase in the metric (increment) between the treatment and control group is computed in the third column. The results of the A/B test are statistically significant or flat depending on whether the spread contains zero or not. We can concretely say for transaction and units that we achieved statistically significant result for Chicago and Houston markets at 95% confidence interval. Overall, we observe a lift in excess of 11% for both units sold and number of transactions with all of our sample, well exceeding the baseline resolution of 2.5%.

6 CONCLUSION AND FUTURE WORK

In this work, we studied the problem of product substitution and its specificity to the shopping channel. We discussed a few potential approaches and their relative performance in providing online grocery product substitutes recommendation to the customers. We found that GraphSWAG, a graph convolutional approach, that combines information from store sales as well as product description outperforms approaches relying solely on store sales or product description. We leveraged the product substitution work to factor in demand transfer of unavailable products to existing assortment and hence plan assortment of cookies and crackers in stores. Significant lift was observed in units sold as well as number of customer transactions. As next steps, we would like to scale the assortment planning effort to more categories of groceries, as well as use graphSWAG based substitutions to facilitate assortment planning. We also demonstrated the relative performance of these approaches in providing out of stock recommendations on the website as well as in same-day delivery scenarios.

ACKNOWLEDGEMENTS

The authors would like to thank Bharath Rangarajan, Elif Erdemir, Jacob Portnoy, Ravi Pandey, Ramasubbu Venkatesh and Anna Holschuh for their insights and help in shaping this project.

REFERENCES

[1] https://progressivegrocer.com/how-digital-transformation-reshaping-grocery-industry. *last accessed 26 Jan 2020.*
[2] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.
[3] A Brooke, D Kendrick, A Meeraus, and R Raman. Gams/cplex 7.0 user notes. *GAMS Development Corp*, 2000.
[4] M. Charikar. Similarity estimation techniques from rounding algorithms. *ACM Symposium on Theory of Computing*, 2002.
[5] Marshall L. Fisher. OR FORUM-rocket science retailing: The 2006 Philip McCord Morse lecture. *Oper. Res.*, 57:527–540, 2009.
[6] Tim Head, Louppe MechCoder, Iaroslav Shcherbatyi, et al. scikit-optimize/scikit-optimize: v0. 5.2, 2018.
[7] Houdong Hu, Yan Wang, Linjun Yang, Pavel Komlev, Li Huang, Xi Stephen Chen, Jiapei Huang, Ye Wu, Meenaz Merchant, and Arun Sacheti. Web-scale responsive visual search at Bing. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 359–367. ACM, 2018.
[8] A. Gürhan Kök, Marshall L. Fisher, and Ramnath Vaidyanathan. *Assortment Planning: Review of Literature and Industry Practice*, pages 175–236. Springer US, Boston, MA, 2015.
[9] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.
[10] Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 785–794, New York, NY, USA, 2015. Association for Computing Machinery.
[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA, 2013. Curran Associates Inc.
[12] Amit Pande, Kai Ni, and Venkataramani Kini. Swag: Item recommendations using convolutions on weighted graphs. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1095–1100. IEEE, 2017.
[13] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.
[14] Pang-ning Tan and Vipin Kumar. Mining indirect associations in web data. In *Proc of WebKDD 2001: Mining Log Data Across All Customer TouchPoints*, 2001.
[15] Stephane Tuffery. *Data Mining and Statistics for Decision Making*. Wiley Publishing, 1st edition, 2011.
[16] Fan Yang, Ajinkya Kale, Yury Bubnov, Leon Stein, Qiaosong Wang, Hadi Kiapour, and Robinson Piramuthu. Visual search at eBay. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2101–2110. ACM, 2017.