Retail Based Cost Reverse Engineering and Cost comparison within Item Similarity Clusters for Cost Negotiations

Mani Kanteswara Rao Garlapati[*], Souradip Chakraborty[*]

[*] Data & Analytics, Walmart Labs, Bangalore

Abstract- Detection of negotiable items and performing various cost negotiation strategies is an age-old problem in the field of retail but despite of that, there are lot of areas that are unexplored in the above field.

This paper talks about detection of negotiable items and how cost negotiations can be performed by leveraging retail price-based cost reverse engineering negotiation methods. Retail price change of an item over time is analyzed along with the cost trend of the item to determine potential for cost negotiations. These comparisons are performed within different item similarity clusters, so that we can see how different peer items within the same similarity cluster perform with respect to retail price and cost price trends. Hence our methodology efficiently captures and detects the most robust set of negotiable items considering major attributes including margin percent, sales volume, substitutability etc.

Index Terms- Item substitutes, Community detection, Parent supplier, Dis-intermediation, Joint buy.

I. INTRODUCTION

This paper guides us on Cost negotiations of different items using different negotiation levers to be identified during the process. In this paper we would focus on cost negotiation using one such lever which is based on retail price-based cost reverse engineering. Retail price of any item usually consists of both promotional and non-promotional sales. For performing cost negotiations, we should be able to identify the trend of regular retail price of the item over time which would be obtained after removing all the promotional events. The regular retail price of an item can be computed by removing different promotional effects like clearance, rollback, price adjustments etc. By understanding the trend of change in regular retail price we can identify if there is an opportunity for cost negotiations by comparing it with the trend of average unit cost for each item using correlation coefficient-based metric. These comparisons can also be performed by comparing the trends among similar item description clusters. These item similarity clusters are obtained by leveraging a combination of Lucene elastic search implementation and community detection algorithm.

Retail price of an item changes constantly based on the demand, competitor pricing and various other factors. Price change of an item for rollback is set centrally for any retail and it gets applicable across all items within certain departments. Clearance and Price adjustments are made by store manager when items are not getting cleared and which remain in the shelf for a long period of time. These are tagged with certain report codes for different types of promotions along with the quantity of units sold at the promotional price that was offered. Use these report codes we will be able to separate regular sales and units using which we can then calculate Regular Average Unit Retail (AUR), which is the sell price of unit quantity. The change in trend of the

regular AUR and its comparison with the trend in the Average Unit Cost (AUC) would help in and efficient negotiation of the item cost.

II. IDENTIFY, RESEARCH AND COLLECT IDEA

There is lot of work happening in this space around how various negotiations can be performed. Economic, behavioural, and software agent perspectives based integrative negotiation gained quite a lot of popularity in the field of electronic ecommerce [1].

There has been work around leveraging retail price trend for negotiation cost of an item. Research has also been done on how to identify similar items based on their item details. Community detection and elastic search has not been leveraged till date for detection of similar items based on their descriptions. But both of these together have not been leveraged for cost negotiations.

We have utilized Walmart items details like descriptions, signage etc. for identifying item similarity clusters. Along with these descriptions we have utilized retail price, promotional activities, cost information, warehouse data, external data like holidays, events, climatic conditions etc. for obtaining the regular retail trend and cost price trends for different items across departments.

III. STUDIES AND FINDINGS

Regular retail price of an item refers to non-promotional price over time. We can build out data at week level in order to avoid day level nuances in the data and avoid data sparsity for low selling items. Also, we are averaging the prices of different items across different stores. This help in generalizing the retail and cost data trend over time and across stores. We will need to compute actual sales, units across different weeks. Flag different promotional events basis their report codes from the data to identify the promotional contribution of various events for their sales and units. Regular sales and units can be computed using the below formula.

Regular Sales = Actual Sales – Rollback Sales – Clearance Sales – Price Adjustment Sales

Regular Units = Actual Units – Rollback Units – Clearance Units – Price Adjustment Units

Regular Average Unit Retail (AUR) can be computed as a ratio of Regular Sales and Regular units over different weeks. Similarly, Average Unit Cost (AUC) can be computed as a ratio of Net ship cost to the Net ship quantity over different weeks.

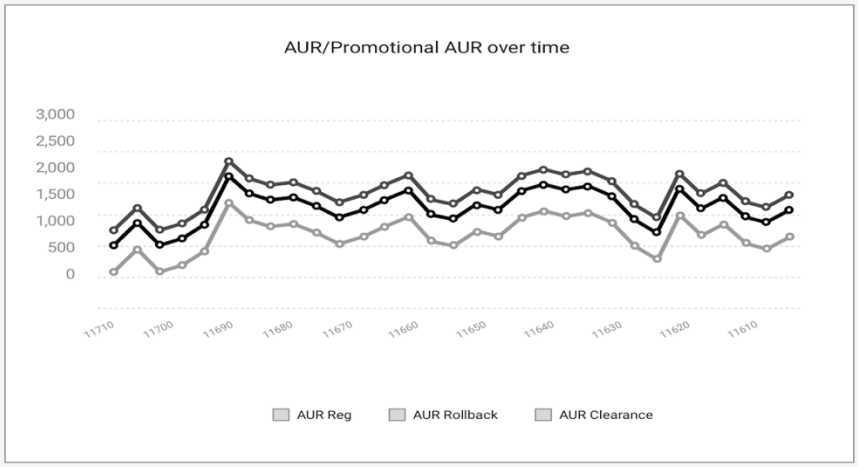

Figure 1: Promotional Breakdown of Retail Price

On computation of AUR and AUC separating out the promotional effects, the trend of the variables has been compared using Pearson's correlation coefficient ratio metric.

$$r\_ratio = (r_{AUR,time}) / (r_{AUR,time} + \varepsilon) \quad (1)$$

where,

r_ratio - correlation coefficient ratio metric

$r_{AUR,time}$ – correlation coefficient of AUR over time

$r_{AUR,time}$ - correlation coefficient of AUC over time

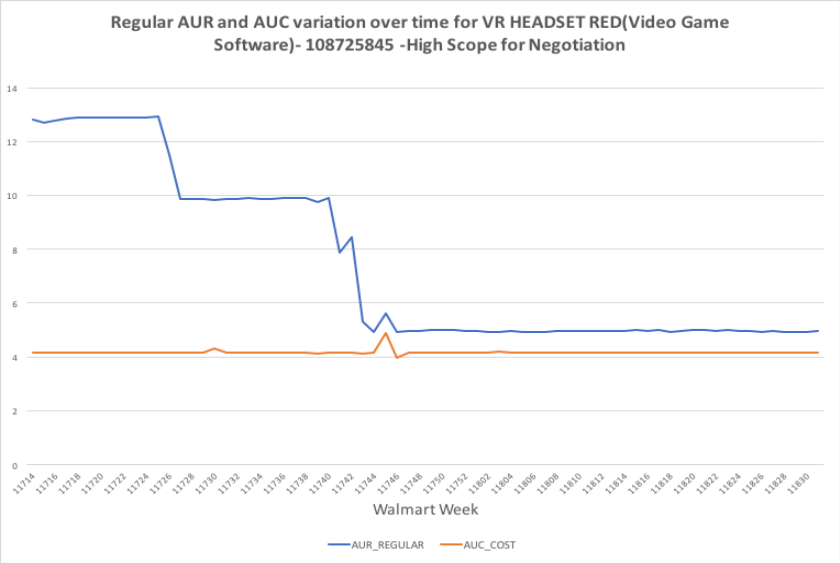

Figure 2: Trend based analysis of AUC and AUR

The correlation coefficient gives an adept understanding of the variation of the one of the variables with respect to another and hence in this case it has proven to be efficient in detecting negotiable items efficiently.

The entire scenario can be broken down into the following cases:

1. Sale Price increasing, Cost Price increasing
    i. r_ratio > 1 - No action needed
    ii. r_ratio <1 - Scope for Negotiation
2. Sale Price decreasing, Cost Price decreasing
    i. r_ratio > 1 - Scope for Negotiation
    ii. r_ratio <1 - No action needed
3. Sale Price decreasing, Cost Price decreasing
    i. No actions needed
4. Sale Price decreasing, Cost Price increasing
    i. Scope for Negotiation

As shown in Figure2, the AUR retail price decreased with time for that item whereas the unit cost price remains almost constant giving an indication for potential negotiation.
Comparison of Average unit cost (Cost per unit) across different items within similar item clusters can also aid to cost negotiation. Our aim is to find items with similar description and create clusters, so that we can perform comparison of AUC among the items within the same cluster. Also, these clusters can be leveraged for additional cost negotiation opportunities such as Join Buy, Dis-intermediation and Parent-Supplier Connection.

For creation of item similarity clusters, we need first obtain similar description-based items. It is computationally expensive as we need to find similarity score between all the possible item pairs. Product description of that item includes the most relevant information of the same including its textual details, category, subcategory, fineline information,color, texture, brand etc.
To compute the pairwise item similarity, we have extracted various natural language based features including n-grams, percentage match, words share etc. The metrics thus formed have been used to compute various similarity scores like Jaccard index, Cosine similarity etc. whose weighted average gives the separation between the two items which is an indication of item substitutability.

To make the number of item pairs for computing similarity metrics, we have leveraged Lucene search library [9]. Lucene is an inverted full-text index. It takes all the documents, splits them into words and then builds an index for each word. We would take help of the item descriptions like signage, upc, supplier inputs etc. to create the index for the items. Lucene library is utilized for building the index on the concatenated descriptions of the item. Once we have the index built, we parse each item through the index to obtain top 50 most relevant similar items based on their descriptions. The resultant set would be up to 50 most relevant items for each such item that has been parsed based on the similarity score computed by Lucene. This process continues for all the items that have been considered. The output is in the form of similar item pairs with their scores. The flow for computing these similar cluster items can be shown in Figure 3.

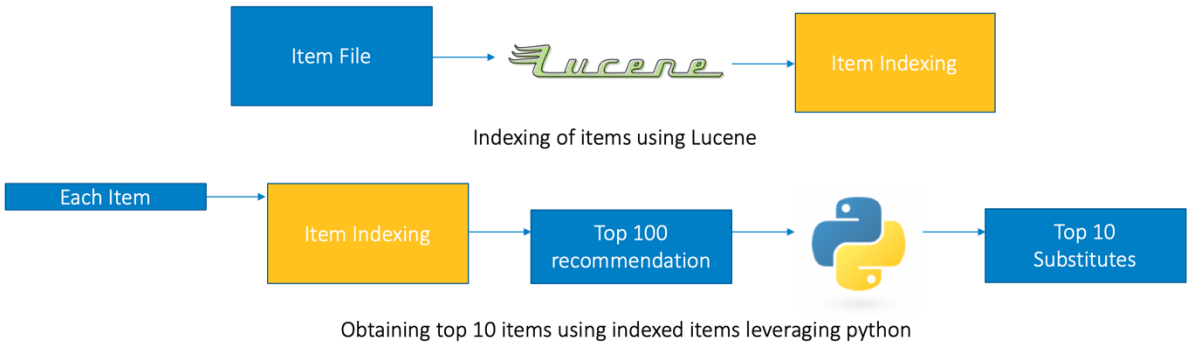

Figure 3: Flow for Computing similar cluster items

The similar item pairs obtained using the Lucene search functionality can be visualized in the form of a network. Each item would resemble a node and every item pair resembles an edge connecting the nodes as shown in Figure3. The scores between each item pair is utilized as edge weight in the graph. We need to identify communities of similar items among these item pairs..

There are numerous algorithms to achieve this, but we need to choose an algorithm which can follow top-down approach. Firstly, lets understand how different algorithms operates.

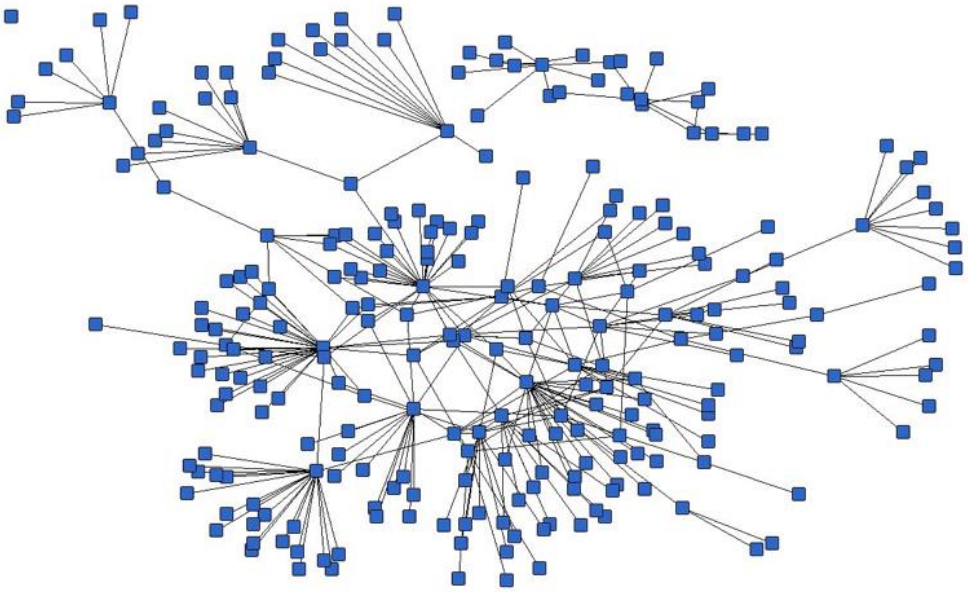

Figure 4: Community Detection in Graphical Networks

Edge-betweenness-community is a hierarchical decomposition method where edges are removed in the decreasing order of their edge betweenness. This is done assuming that edges connecting different groups mostly contain multiple shortest paths. It yields good results but is slow since its computation intensive due to calculations of edge betweenness. Also, they need to be re-calculated after every edge removal. It's ideal for few thousands of vertices in the data. It builds a full dendrogram and does not give the final cut off point to obtain the final groups.

Fast greedy-community is another hierarchical approach, but it is bottom-up instead of top-down. Modularity gets optimized in a greedy manner. Communities are merged iteratively considering every vertex belongs to a different community so that each merge is locally optimal. The algorithm stops when modularity has reached its saturation. The method is fast and hence most sorted as there are no parameters that needs to be fine-tuned. One drawback is that communities below a given size gets merged with neighboring communities.

Walk trap-community performs random walks on the network. These walks tend to stay within the same group because there would be few edges that lead to outside group. Walk trap runs short random walks of 3-5 steps and results are used to merge separate communities in a bottom-up fashion. Modularity score can be leveraged to select where to cut the dendrogram. It is a bit slower than the fast-greedy approach but is more accurate.

Spinglass-community is from statistical physics based on Potts model. Each vertex of the network can be in one of c spin states, and the edges of the specify which pairs of vertices would prefer to stay in the same spin state. Communities get defined after simulating for a given number of steps based on the spin states of the vertices in the network. This method is not fast and not deterministic due to the inherent simulation, but has a tunable parameter to determine the cluster size.

Leading-eigenvector-community is a top-down hierarchical method which optimizes the modularity function. At each iteration the network is split into two parts such that the separation yields a significant increase in modularity. Split is determined by evaluating leading eigenvector of modularity matrix. Due to the eigenvector calculations, it might not work on degenerate networks.

Label-propagation-community is a simple method where every node is assigned one of k labels. During different iterations it re-assigns labels to nodes in a way that each node takes the most frequent label of its neighbors. The method stops when the label of each node is one of the most frequent labels in its neighborhood. It is very fast but yields different results based on the initial seeds. Hence, one should run the algorithm a large number of times and then build the final communities which is tedious.

For this paper we have utilized customized constraint-based Louvain community detection [10] algorithm to determine the optimal clusters. This algorithm also optimizes modularity which is a scale value between -1 and 1 that measures the density of edges inside communities to edges outside communities. First small communities are identified by optimizing modularity locally on all nodes, then they are grouped into one node for the next iteration.

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j), \tag{2}$$

where,

- A_{ij} represents the edge weight between nodes i and j .
- k_i and k_j are sum of weights of the edges attached to nodes i and j
- $2m$ is the sum of edge weights in the graph
- c_i and c_j are communities of the nodes
- δ is a delta function

The modularity (2) has been maximized using constrained optimization on the UPC information.

To identify how granular the communities needs to be formed is a critical task. For solving this problem, we have leveraged the UPC information. The same UPC might be supplied by different suppliers and would have different item numbers. Since we are grouping different item numbers under the communities, the UPC information would help us to determine if the communities have been formed correctly. We need to ensure that most of the communities that have been formed contains all the items of the same UPC number falling into the same community. If this fails for 90% of the items then we re-run the algorithm by fine tune the edges and breaking the communities further to achieve the above constraint.
On removing weak edges from the network, the Louvain algorithm would create more granular clusters leading for most of the items having same UPC to fall into the same community. Several iterations are performed and under each iteration a new network is created with the fine-tuned edges and the communities are identified ensuring that 90% of the items having same UPC fall into the same community.

Once communities are identified and all the items are tagged to each of the community. We would compute the volume and margin percent for each of the item. Basis the distribution of the volume and margin percent we are going to classify each of the cluster into four groups as shown in below table. Each group would help every retailer to take a strategic decision for improved quality and sales as shown in Table 1.

Table 1	
Group	**Strategic Decision for Retailers**
High Selling High Margin	Strategic Items
High Selling Low Margin	Negotiation Opportunity
Low Selling High Margin	Marketing/Promotional Opportunity
Low Selling Low Margin	Alternate Sourcing Opportunity

Table 1: Shows how retailers can take strategic decisions using the Group information different items get assigned

Also, retail-based cost negotiations can be performed within each cluster that has been formed. For example, in the below cluster we can see that there are 4 similar items. But one can observe that we have a good margin percent and good volume percent for the first two items on comparison with the bottom two items. Customers are tending to buy smaller packs comparative to the larger packs in this variety of items. As a retailer we can improve the profits by procuring more quantities of first two items as they have been liked by the customers and they have been having maximum margin percentage as shown in Table 2.

Table 2		
Item Name	**Units%**	**Margin%**
GV NFC ORANGE JUICE 59OZ CARAFE	0.69	0.29
GV NFC ORANGE JUICE WITH PULP 59OZ CARAF	0.36	0.29
GV NFC ORANGE JUICE 109OZ CARAFE	0.01	0.04
GV NFC ORANGE JUICE WITH PULP 109OZ CARAF	0.01	0.07

Table 2: Shows items falling in same cluster for cost negotiations

All the item clusters that have been formed would be classified to one of the groups as shown in the table 3. Each group would help to take a strategic decision by the retailer which would eventual help in increased customer satisfaction, increased sales and more footfalls.

Table 3		
Cluster No	**Item Name**	**Group**
1	MYBIOME WOMENS PROBIOTIC	Low Selling High Margin
1	MYBIOME MENS PROBIOTIC	Low Selling High Margin
1	MYBIOME 50+ PROBIOTIC	Low Selling High Margin
2	OLE CREMA SQ 15OZ	High Selling High Margin
2	OLE CREMA SQUEEZE	High Selling High Margin
3	SIMPLY GRAPEFRUIT JUICE 52OZ	High Selling Low Margin
3	SIMPLY CRANBERRY COCKTAIL 52OZ	High Selling Low Margin
3	SIMPLY GRAPEFRUIT JUICE	High Selling Low Margin
4	TROPICANA PP LEMONADE 52OZ	Low Selling Low Margin
4	TROPICANA PP RASPBERRY LEMONADE 52OZ	Low Selling Low Margin
4	TROPICANA PP PEACH LEMONADE 52OZ	Low Selling Low Margin

Table 3: Shows sample clusters number and the groups they have been assigned

IV. CONCLUSION

Analyzing just average unit retail and comparing with the average unit cost would help in identifying cost negotiation possibility in one dimension. On identification of item similarity clusters and comparing AUC with AUR across different items within the same item similarity cluster would help in leveraging cost negotiation opportunities in multiple dimensions, since item cost performance across different items is being compared. Tagging of each item into different groups in terms of their selling and margin performance helps the sourcing managers to better negotiate with different suppliers across the globe.

ACKNOWLEDGMENT

We want to grasp this opportunity to convey our deepest gratitude to all of them who aided us in completing our paper. Our heartfelt gratitude to AICAAM team, for this opportunity. Our great thankfulness to Data and Analytics Team, Walmart International, for introducing us to this topic and guiding us throughout our paper. A special thanks to Ashish Gupta, Manager Walmart Labs as our internal guide of the paper, for his persistent encouragement and motivation towards us. We are thankful to all of them who directly and indirectly helped us in accomplishing this paper.

REFERENCES

[1] Robert H. Guttman,Pattie Maes, "Agent-Mediated Integrative Negotiation for Retail Electronic Commerce", in *AMET*, 1998, pp 70-90.

[2] Lewicki, D. Saunders, and J. Minton. Essentials of Negotiation. Irwin, 1997.

[3] Rosenschein and G. Zlotkin. Rules of Encounter: Designing Conventions for Automated Negotiation among Computers. MIT Press, 1994.

[4] C. Beam, A. Segev, and J. G. Shanthikumar. „Electronic Negotiation through Internet-based Auctions." CITM Working Paper 96-WP-1019, December 1996.

[5] E. H. Miller, "A note on reflector arrays (Periodical style—Accepted for publication)," *IEEE Trans. Antennas Propagat.*, to be published

[6] C. Sierra, P. Faratin, and N. Jennings. "A Service-Oriented Negotiation Model Between Autonomous Agents." *Proceedings of the Eighth European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW'97)*. Ronneby, Sweden, May 1997.

[7] Kraus, S.: Negotiation and Cooperation in Multi-Agent Environments. Artificial Intelligence journal, Special Issue on Economic Principles of Multi-Agent Systems, 94(1–2):79–98, (1997).

[8] De Paula, G. E., Ramos, F. S., Ramalho, G. L.. Electronic Commerce Negotiation Model Formalization. Technical report UFPE-DI 99-20. 1999

[9] Balipa, Mamatha & Ramasamy, Balasubramani. (2015). Search Engine using Apache Lucene. International Journal of Computer Applications. 127. 27-30. 10.5120/ijca2015906476.

[10] De Meo, Pasquale & Ferrara, Emilio & Fiumara, Giacomo & Provetti, Alessandro. (2011). Generalized Louvain Method for Community Detection in Large Networks. International Conference on Intelligent Systems Design and Applications, ISDA. 10.1109/ISDA.2011.6121636.

[11] Blondel, Vincent D; Guillaume, Jean-Loup; Lambiotte, Renaud; Lefebvre, Etienne (9 October 2008). "Fast unfolding of communities in large networks". *Journal of Statistical Mechanics*: Theory and Experiment. 2008