Team: Hacktailers

Members:
Nehal Dhande
Samruddhi Padture
Tejashree Date

Case 2: Spark innovation with a robust Wal-Ket basket.

Proposed solution:

The solution to the Wal-Ket basket is divided in two parts:

1. Batch the user orders in groups of three
2. Find the optimal path for the associate to collect the items.

- Batching the orders:
  Assuming that the input provided to the Wal-Ket basket consists of orders from 10 users, we batch 3 orders together based on their aisles.
  - Map the product_id to corresponding aisle
  - Create a string-like object for each user with unique aisle numbers.
  - Find similar aisles to group orders together that will reduce associate's time.

  Method to be used: Levenshtein distance

  The Levenshtein distance is a string metric for measuring the difference between two sequences. The Levenshtein distance between two words is the minimum number of single-character edits required to change one word into the other. It is also known as "edit" distance.

Algorithm:

1. Create an NxN matrix, N = the total number of orders.
2. Create an array called "done" to mark the orders which have been already processed.
3. Mark the diagonal as 9999.
4. Compute the edit distance pairwise and select top 2 values in a column with minimum edit distance. After selecting the orders mark them as visited in "done" array.
5. Repeat step 4 until the last user in the list is marked as visited.

Example:

Product 123: Aisle A1, Product 124: Aisle A2, Product 125: Aisle B3, Product 126: Aisle D2

Suppose User 1 has ordered Product 123: Quantity-4, Product 124: Quantity-2, Product 125: Quantity-1

[A1, A1, A1, A1, A2, A2, B3]
Will be converted to [A, B] (Meaning that the associate has to visit only 2 aisles to collect all the products required by user1)

Similary, Assume the following strings for all users. (Considering only 4 aisles right now for simplicity)

User 1: [A, B]
User 2: [A, D]
User 3: [A, B, C, D]
User 4: [A, B, C]
User 5: [A, B]
User 6: [A, C]
User 7: [A, D]
User 8: [B, D]
User 9: [B, C]
User 10: [B, C, D]

According to the algorithm,
We create a matrix (10x10) and assign 9999 to diagonal values. We start calculating edit distance column wise.

We also create a “done” array where we insert orders as they’re batched.

	AB	AD	ABC D	ABC	AB	AC	AD	BD	BC	BCD
AB	9999									
AD	1	9999								
ABC D	2		9999							
ABC	1			9999						
AB	0				9999					
AC	1					9999				
AD	1						9999			
BD	1							9999		
BC	1								9999	
BCD	2									9999

We then select top 2 orders with minimum distance and hence get the three orders that are batched together.

Batch 1: [AB, AD, ABC]

We mark these 3 orders to “done’ array and assign 9999 to these rows and columns, as we no longer need to process them. We do this for all the columns.

The matrix will look like this, after all orders are processed.

	AB	AD	**ABCD**	ABC	AB	**AC**	AD	BD	BC	BCD
AB	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999
AD	**1**	9999	9999	9999	9999	9999	9999	9999	9999	9999
ABCD	2	9999	9999	9999	9999	9999	9999	9999	9999	9999
ABC	**1**	9999	9999	9999	9999	9999	9999	9999	9999	9999
AB	0	9999	**2**	9999	9999	9999	9999	9999	9999	9999
AC	1	9999	2	9999	9999	9999	9999		9999	9999
AD	1	9999	2	9999	9999	**1**	9999	9999	9999	9999
BD	1	9999	2	9999	9999	2	9999	9999	9999	9999
BC	1	9999	2	9999	9999	**1**	9999	9999	9999	9999
BCD	2	9999	**1**	9999	9999	9999	9999	9999	9999	9999

Batch 1: [AB, AD, ABC]
Batch 2: [ABCD, AB, BCD]
Batch 3: [AC, AD, BC]

- **Finding Optimal Path:**

  In order to find the optimal path we have two inputs viz the array consisting of the list of items (aisles) and the floor map of the store. The map can be represented in the form of a grid where every grid square contains a unique pair of (x, y) co-ordinate.

  The algorithm can be viewed into two steps/layers:

  1. The first step will be to find the order/sequence of the grocery items (aisles) to be visited. The problem can be viewed as a Travelling Salesman Problem (TSP), where the starting point is the entrance and the cities are the grocery items in the shelves (aisles) to be visited. The objective of the algorithm is to solve the travelling salesman problem (TSP) to get the order in which the nodes have to be traversed.

  2. The second step of the algorithm will be to find the shortest path from item to item using Dijkstra's algorithm. The output of the first step will serve as the input to this step.