WEIGHTED BASKET ALGORITHM

This algorithm works like weighted average methods but for each user-item pair. A weightage is calculated for all those products that the user has interacted with thus stating its importance for the user. All the previous order_ids are taken into consideration while calculating this weightage. The computation is done as follows:

1. A list is made of all the products the user has interacted with.
2. If the product exists in an order, weightage is given according to the order_number. If the product was not purchased in certain orders, then 0 weight is assigned for that order.
3. The total weightage is then summed up and divided by the total weightage to give the weighted score of the product.

Example: Consider a **USER X with 5 Orders and 5 Interacted Products.**

USER X	Order1 Weight = 1	Order2 Weight = 2	Order3 Weight = 3	Order4 Weight = 4	Order5 Weight = 5	Weightage Total = 15
Product1	1	2	3	4	5	15/15=1
Product2	1	0	3	4	5	13/15=0.86
Product3	0	0	3	4	5	12/15=0.80
Product4	0	2	3	4	5	14/15=0.93
Product5	0	2	0	4	5	11/15=0.73

- The most recent orders are given higher weightage as the are more important in predicting relevant products for a user.
- Product 1 is extremely important to the user as it is purchased in every order. Thus, its score is 1.
- Product 2 & 4 are purchased the same number of times ie 4 out of 5, but the baskets differ thus their scores also change. Product 4 is more important as it is present in the most recent baskets as compared to Product 2.
- Product 3 & 5 are purchased the same number of times ie 3 out of 5, but the baskets differ thus their scores also change. Product 3 is more important as it is present in the most recent baskets as compared to Product 5.

This method helps in understanding the user's preferred products over time. As this computation is done individually for every user as the number of orders and interacted products differ for everyone, the process becomes user-centric and personalised.

PRODUCT TRANSITION PROBABILITY

This algorithm calculates reorder probabilities by creating a Markov Chain for each product. One way to model the system is by using a state that indicates whether or not a product was included in the last order; i.e., the state at stage k is

Xk = {1, if the product is in the current order

= {0, otherwise

This is what the transitions of the system look like:

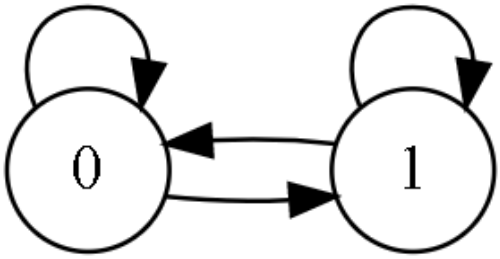

Fig No 6: System Transition Graph

Here a transition from 0 to 1 would mean that a product was not included in the previous order but was included in the current order. Similarly, a self-transition from 1 to 1 would mean that a product was included in both the previous and current order. We are interested in calculating the conditional probabilities of transitioning from state to state. The simplifying assumption behind Markov Chains is that given the current state, the next state is independent of its history

$$Pr[xk+1=x|xk, xk-1,...,x0] = Pr[xk+1=x|xk]$$

and so we can calculate:

P00 =Pr[xk+1=0|xk=0] // transition out of state 0 and into state 0

P01=Pr[xk+1=0|xk=1] // transition out of state 0 and into state 1

P10=Pr[xk+1=1|xk=0] // transition out of state 1 and into state 0

P11=Pr[xk+1=1|xk=1] // transition out of state 1 and into state 1

But the weighted score that is available from the previous algorithm states that the product is included in the previous order. Thus, we need to find what is the probability that the product will be present in the next order given that the product was present in the past order i.e. P11.

P11 is calculated for all the products individually. To calculate P11, all the orders are compared with their previous orders to check if the product has reoccurred in the next order. This can be done by comparing a user's current order with its previous order by the same user. The process needs to be done for every order available in the data.

$$P11 = \frac{\text{No.of times the product was present in both current and previous orders}}{\text{No.of times the product was present in the previous orders}}$$

P11 gives the overall product transition probability of a product i.e. probability of the product reoccurring in the next order given that it was purchased in the previous order. This probability score is product-centric and not specific to a user.

COMBINING WEIGHTED BASKET AND PRODUCT TRANSITION PROBABILITY

The weighted basket algorithm gives the weighted score of all the user interacted products thus calculating the importance of a product compared to the other products in that user's list.

While product transition probability gives the probability of the product reoccurring in the next orders.

If these two algorithms are combined they would give a score that will not only be user-centric(due to Weighted Basket) but also take into consideration the generic behaviour if the products(due to Transition Probability).

Thus, the weighted average score is multiplied with the transition probability of those products. Now every user has a list of their interacted products with a score assigned to it. This list needs to be sorted for products with the highest score at the top. Every user will not only have unique products on their list but also the number of products will differ.

The number of recommended products cannot be different for each user as the recommendation spots(places where the recommendations are given on the website) be limited and similar every time, this list needs to be filtered. Thus, the top 10 products are filtered out for each user to create a personalised recommendation list. This number can be altered according to the available recommendation spots.