

Scalable and Personalized Item Recommendations Framework

Nimesh Sinha
WalmartLabs
Nimesh.Sinha@walmartlabs.com

Selene Xu
WalmartLabs
yue.xu@walmartlabs.com

Swati Bhatt
WalmartLabs
Swati.Bhatt@walmartlabs.com

Abhinav Mathur
WalmartLabs
Abhinav.Mathur@samsclub.com

Jason H.D. Cho
WalmartLabs
hcho@walmartlabs.com

Sushant Kumar
WalmartLabs
SKumar4@walmartlabs.com

Kannan Achan
WalmartLabs
KAChan@walmartlabs.com

ABSTRACT

Balancing scalability and relevance is important for industry-level recommender systems. A large scale e-commerce website may have millions of customers and millions of items. Typically, researchers and data scientists generate models which are anchored on customer level, or item level. Customer anchored recommendation models are typically surfaced on front pages of e-commerce websites. Similarly, item pages typically host various item anchored models. In each of the two usecases, developers frequently store models offline, i.e., models are stored for each customer, or are stored for each item. Scalability challenges arise when one wishes to personalize item anchored models. Offline based approaches, where both customer and item ids are stored as anchors become rapidly unscalable as number of customers and items increase. Another approach is to utilize an online approach on a pre-computed recall set (for example: item recommendations for a given anchor item), and then to apply users' preferences. In this paper, we describe a scalable personalized item recommender system which follows the latter approach. We take historical user preferences (customer understanding) and existing item recommendation models to personalize item anchored model. We showcase several usecases to show how we apply online inferencing algorithms and scale it up to millions of customers.

CCS CONCEPTS

• **Information Systems** → *Recommender Systems*.

KEYWORDS

Personalization, Recommender systems, Scalability, Customer Understanding

ACM Reference Format:

Nimesh Sinha, Selene Xu, Swati Bhatt, Abhinav Mathur, Jason H.D. Cho, Sushant Kumar, and Kannan Achan. 2020. Scalable and Personalized Item

Recommendations Framework. In ., ACM, New York, NY, USA, 6 pages.
<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 INTRODUCTION

Recommending personalized items is an essential but challenging task for e-commerce. Personalized recommendations provide customers with more relevant suggestions, hence improving customer experience. One challenge that personalized recommendation system faces is scalability. This is because such system must be anchored on combinations of customer and item and the size of such storage schema can get aggressively large. This is especially true for large e-commerce sites such as walmart.com where millions of users interact with millions of products daily on the site. We come up with a novel re-ranker framework where item relevance and customer preference act as independent and parallel factors. This structure allows us to break down (customer id, item id) anchored storage into (customer id) anchored storage and (item id) anchored storage, thus reducing storage requirement. Moreover, on the modeling side, we now have the freedom to train item relevance models and customer preference models separately. The re-ranker will eventually combine the results from both models using a machine learned inference function and provide a final list of recommendations.

A main motivation for our re-ranker system is that there are often existing non-personalized item anchored recommendation models that show great performances. These models have been fine tuned through multiple iterations and provide a very good recall set. We do not want these work to go to waste and start building personalized system from scratch. So we design our re-ranker to utilize the results from these non-personalized models as our recall set and implement personalization within the established recall set. Our personalization models target different attributes of personal preference such as price, brand, flavor, and etc. The re-ranker then uses these attribute preferences as independent features. This allows our data scientists the freedom to focus on personalizing a single attribute at a time instead of having to reformulate existing item anchored recommendation models. Our system can also rapidly extend to different activations (e.g. similar item, complementary item, etc.) because it is able to utilize the same attribute features and similar inference function structure across different activations. The only change that needs to happen from activation to activation is to re-calculate/re-balance the weights between item relevance

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2020 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00
<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

and personal preference as well as the weights associated with each attribute feature.

2 RELATED WORKS

There has been a plethora of research in the field of personalized recommendation systems. The traditional matrix factorization model [6] has been extensively studied and many of its variants have been developed to tackle different challenges such as implicit feedback and sequential prediction [5] [11] [12]. More recently, many embedding methods such as word2vec [8] [9] have been extended to the e-commerce domain [1] [13] [14]. Most recently, graph based methods have gained much traction in personalized recommendation. Many state-of-the-art recommender systems are designed using tools such as Graph Convolutional Neural Networks [2] [17] and Knowledge Graphs [4] [16].

However, not many research discuss how to actually implement their models at scale. Of note, representation learning methods have been widely used to support large-scale recommendation engine [3] [7] [10] [15]. The difference between these works and ours is that they model implicit customer preferences through user past behavior whereas we try to generate explicit user understanding.

3 SYSTEM ARCHITECTURE

Here at Walmart, we maintain both offline item anchored recommendation and customer understanding models. We describe in detail how the inference function utilizes both the existing item anchored recommendation model and customer preferences to provide personalized item anchored recommendation. We then detail on how we utilize the inference function on the serving layer.

3.1 Inference Function

The goal of the inference function is to identify, for a given user and item, what is the most optimal set of recommendation. More precisely, we optimize

$$\max_{r \in I} \text{score}(u, i, r) \quad \forall u \in U, i \in I \quad (1)$$

where u is user, i is anchor item, and r is set of possible recommendations. Notice that, while the aforementioned formulation is still tractable, in an industry setting, it is difficult to pre-compute the scores. At Walmart, we have tens of millions of customers, along with tens of millions of items, so computing the best (u, i, r) triplet becomes very difficult.

Rather, we take an existing item recommendations, and frame it as a re-ranking framework. More precisely, suppose that we have an existing item recommendation which computes

$$R_i = \max_{r \in I} \text{score}(i, r) \quad \forall i \in I \quad (2)$$

where R_i is the set of recommendations for anchor item i . Now the optimization function becomes

$$\max_{r \in R_i} \text{score}(u, i, r) \quad \forall u \in U, i \in I \quad (3)$$

where r is limited to the set R_i .

Notice that the above formulation still suffers from having to pre-compute recommendations for each (u, i) . To mitigate this issue, we treat $\text{score}(u, i, r)$ as a combination of two different products. We first notice that we do not need to predict which anchor item

i a given user u is likely to view or purchase next. On the other hand, we do need to understand a users' preference towards a recommended item $r \in R_i$. We denote the users' preference towards a recommended item as $g(u, r)$. We compute $g(u, r)$ by utilizing customer understanding which we will illustrate in the case study.

Furthermore, we have an existing item anchored model, $h(i, r)$. These models are optimized based on which recommended items $r \in R$ best optimize a given business metric. Examples of these include With $g(u, r)$, and $h(i, r)$, we are able to compute

$$\text{score}(u, i, r) \approx f(g(u, r), h(i, r)) \quad (4)$$

The benefit of the above representation is the decoupling between item-anchored model optimization (denoted by $h(i, r)$), and that of customer preference model ($g(u, r)$). Such decoupling allows the two specialized track to work in parallel, and then to mutually work on a function $f()$ which binds the two models together.

3.2 Overall Architecture

Our training infrastructure is set up in such a way that customer understanding track, and item recommendation (item anchored model) track can work independently of one another. Once each of the tracks have trained their model, these are pushed to an existing database. Our system can start training once the two tracks are pushed to the database.

Our proposed re-ranker system works for all the different activation. For each activation, we train the inference function independently using a supervised machine learning algorithm. We use the best performing item anchored model to get the relevant scores. Our framework is not dependent on this and we can use the scores coming from any kind of model. We utilize the customer understanding attribute scores along with the relevance scores for training the inference function.

We use the customer historical activity data depending on the activation for training the weights for relevance score and each customer understanding attribute - brand, price, flavor, etc. Our re-ranker system has the flexibility of selecting any number of customer attributes from the available list of attributes. These weights can be present at global level or any taxonomy hierarchy level for the contextual information. We collect the customer past module related feedback data and create relevant labels. Then, we train the weights offline optimizing different function for different activation and get weights for relevance and each of the customer attributes selected.

We use these weights for offline evaluating the different relevant metrics. Many times, we use the ranking metrics like Normalized Discounted Cumulative Gain(NDCG), Mean Reciprocal rank(MRR), Mean Hit Rate(MHR) and Mean Average Precision(MAP) to compare the baseline model recommendations against the re-ranked model recommendations.

Once we finalize the re-ranking model that outperforms the baseline model, we make the corresponding model weights and inference function available to the serving layer, which uses them for online inferencing and re-ranking the recommendations.

We maintain a pipeline for continuously collecting various useful metrics like percentage of re-rankings happening, coverage of each customer attributes, and the above mentioned ranking metrics. We

have a feedback loop where use the new feedback data to train the model and generate new weights.

Our overall architecture is shown in Figure 1.

4 CASE STUDIES

We demonstrate how our re-ranker works through two case studies, each of which hinges on a distinct track of customer preference understanding. We first provide an overview of how customer understanding models come about. Customers may search, view, click, add-to-cart, or purchase various items at Walmart. Each of these items have meta-information such as brand, flavor, or price-level associated with it. We use these meta-information to build customer preferences. As an example, a customer may decide to view, and ultimately purchase ‘Chobani Non-Fat Peach Greek Yogurt’ over multiple visits to an e-commerce website. The same customer may later on purchase ‘Chobani Non-Fat Strawberry Greek Yogurt’. Based on these interactions, we learn that a customer may prefer the ‘Chobani’ brand, prefers ‘non-fat’ yogurt and likes ‘peach’ and ‘strawberry’ flavors. Here at Walmart, we maintain each customer’s preferences across different attributes such as flavor and brand. We denote these preferences as ‘Customer Understanding’.

In this section, we will showcase how our re-ranker carries out personalization using two different aspects of customer understanding: brand and price. We will apply brand understanding to the “bought also bought” (BAB) application and apply price understanding to the “view ultimately bought” (VUB) application. Both of these applications are currently powered by item anchored modules.

4.1 Brand Understanding

We use a very simple and straightforward tf-idf style method to compute customer brand affinity scores. This score essentially reflects how much a customer has interacted with a given brand in the past. Moreover the customer brand affinity scores are computed at the category level. For example, the latest iphone will fall under Prepaid Cell Phones category. One brand might have products across multiple categories and our customer brand affinity scores can capture a customer’s preference for brands under each specific category.

While our customer brand affinity model provides explicit signals for customer brand preference, it suffers from poor coverage. This is because if a customer has never interacted with a brand before, then we wouldn’t have an explicit brand preference for this person and the brand. In our experiment, customer on average has 14.2% of brand coverage. In order to compensate for this, we develop a brand2brand model that provides implicit signal on customer preference for brands they have not interacted with before. The brand2brand model is based on the word2vec model [8] [9] trained on historical session data of items co-viewed within the same session. Within the word2vec framework, each session on our website is treated as one document and each brand at a specified category level is treated as a word. Some example results from the brand2brand model are shown in Table 1.

For a given brand, the brand2brand model provides a list of the most similar brands. This is useful to back-filling the missing explicit customer brand affinity scores as we can use the following inference logic. When we are missing a customer C’s explicit brand affinity

score for brand A, we go down the list of the most similar brands to brand A. We then use the first brand, say brand B, where we have explicit brand affinity score from customer C to brand B, multiply this known customer brand affinity score with the similarity score between brand A and brand B, and use this product as the customer brand affinity score from customer C to brand A. Applying the brand2brand model with the aforementioned logic improves brand coverage to 60.6%.

After developing the explicit customer brand affinity model supplemented by the brand2brand model as implicit customer brand affinity, we are ready to activate our models on the re-ranker for the bought also bought (BAB) application. The current BAB model is used at the Post Add to Cart page of walmart.com. The BAB model has about 1.8 million anchor items and an average of 30 recommendations for each anchor item ranked by a “relevance” score. The re-ranker combines the relevance score and our customer brand affinity score to re-rank the recommendations for each customer. The combination of relevance score and brand understanding is set to be linear to minimize latency and is trained independently within each different category. The exact form is as follows:

$$y_c = w_0^c + w_1^c \times \text{relevance} + w_2^c \times \text{brand_affinity}$$

where c represents category c . The weights are learned offline through a logistic regression model where the response signal is co-boughtness within the same session. We document the offline evaluation results comparing using the re-ranker to personalize and re-rank BAB recommendations against using the relevance score (non-personalized) alone in Table 2.

Model	Non Personalized BAB	Personalized BAB
NDCG@5	0.031	0.034 *
MHR@5	0.046	0.049 *
MRR@5	0.027	0.029 *
MAP@5	0.032	0.035 *

Table 2: Offline evaluation results applying brand understanding to the “bought also bought” application

There is improvement in all of the ranking metrics for personalized BAB model against the non-personalized one. There is 1% lift in NDCG, 0.7% lift in MRR and 0.9% lift in MAP. 78% and 95% of times re-ranking happens across top 5 and top 30 recommendations respectively. The model weights and the inference function are available to the serving layer which does online re-ranking on the recommendations.

We perform a more deep dive analysis on which of the factors the customers give more weightage to- brand affinity or relevance within each category. We find that for some categories of items, customers prefer brand affinity over relevance, while for others, they prefer relevance over brand affinity. Some of these categories are shown in Table 3.

* indicates statistical significance at $p < 0.05$ compared to baseline

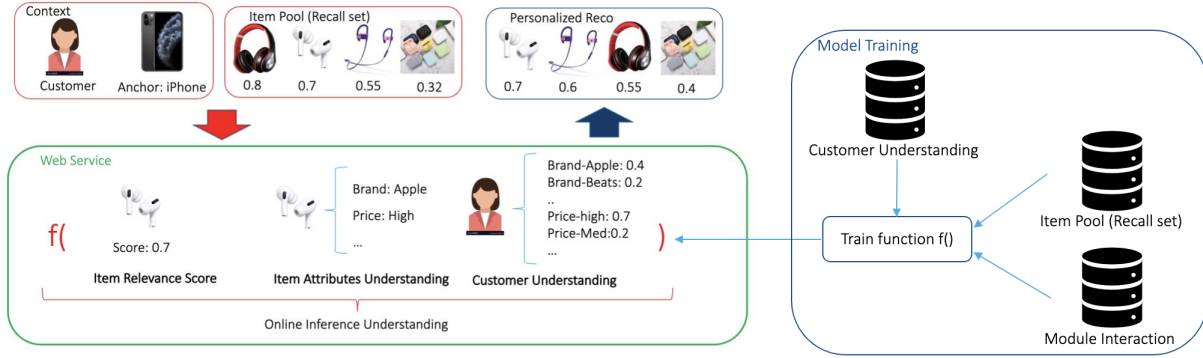


Figure 1: System architecture. Engineers train a function $f()$ by utilizing existing customer understanding, recall set, and historical module interactions for each customer. The serving layer imports the function $f()$ and its model weights.

Brand 1	Category 1	Brand 2	Category 2	Similarity Score
Versace	Women Fragrance	Elizabeth Arden	Women Fragrance	0.96
Versace	Women Fragrance	Coach	Women Fragranc	0.95
Apple	Cellphones	iPhone	Cases	0.90
Apple	Cellphones	Straight Talk	Sim Cards	0.88

Table 1: Example from Brand2brand model

Categories with more brand preference	Categories with more relevance preference
Maternity Sleepwear	Unisex Occupation Footwear
Girls Sweater	Music Gift cards
Girls Woven Top	Floormats
Girls Swimwear	Breakfast and Baking

Table 3: Analysis of brand affinity preference against relevance

We observe that customers have more brand preference over relevance for many of the categories related to clothing. This is understandable because brand is driving distinguisher in the apparel industry. It is hard to generalize for the opposite case where the customers have more preference for relevance over brand. However, other attributes seemed more important in many of the categories in which relevance were higher. As an example, customers may be more concerned with the design or color when they purchase floormats. However, for the purpose of this usecase study, we did not build models based on design or color.

4.2 Price Understanding

In order to understand how price point affects customer’s purchase preference, we first study how price varies from item to item. We assign each item into a price bucket which reflects how expensive the item is compared to its peers. Since we want to compare prices of items of similar property and function, we perform this price banding within each category. We roughly classify items into 5 price bands (low, medium low, medium, medium high, high) within

each category, with lower price band pointing to cheaper/more affordable products and higher price band pointing to more expensive/luxurious options. This item price understanding offers us a simple naive feature for the re-ranker: price2price. This is defined as the difference between the price band of the recommended item and the price band of the anchor item. It measures how far/close the price bands are between the anchor item and the recommended item. Some examples of the price banding results are shown in Figure 2.

On the other hand, we need to understand how customer preference is affected by how expensive a product is. We use a customer’s past behavioral patterns on our website to predict how likely the customer is going to buy something that’s at the price point of the recommended item. We use a machine learning model to determine customer price affinity scores, which will then be fed into the re-ranker as a second feature. Our customer price affinity model covers roughly 41% of all the recommendations.

Ultimately, the re-ranker brings both the customer price affinity feature and the price2price feature into the view ultimately bought (VUB) application. The current VUB model has about 7 million anchor items and an average of 50 recommendations for each anchor item ranked by a "relevance" score. The inference function here is a linear combination of three features: relevance score, customer price affinity score, and price2price:

$$y = w_0 + w_1 \times \text{relevance} + w_2 \times \text{price_affinity} + w_3 \times \text{price2price}$$

The weights associated with each feature here are trained at a global level (as opposed to at each category) and are learned through a logistic regression which optimizes for items viewed and purchased within the same session. The offline evaluation results are shown






Digital Cameras	Low Price	Medium Low Price	Medium Price	Medium High Price	High Price
Item					
Price	\$34.95	\$77.02	\$106.99	\$276.99	\$1,199.00

Figure 2: Examples of products falling under each price band in the Digital Cameras category.

in Table 4.

Model	Non Personalized VUB	Personalized VUB with baseline tf-idf style price affinity	Personalized VUB with machine learned price affinity
NDCG@5	0.294	0.295	0.299 *
MHR@5	0.496	0.498	0.505 *
MRR@5	0.542	0.543	0.542
MAP@5	0.119	0.120	0.122 *

Table 4: Offline evaluation results applying price understanding to the "view ultimately bought" application

In Table 4, we compare the non-personalized model against two personalized models: one baseline method where we use a simple tf-idf approach to compute price affinity scores and one machine learned method where we leverage months of user behavioral data on our website to learn their price preference. While the baseline method slightly outperforms the non-personalized VUB model, the machine learning method clearly proves to be the overall best performer. The machine learning method shows 1.7% improvement in NDCG, 1.8% improvement in MHR, and 2.5% improvement in MAP. The improvements in NDCG, MHR and MAP@5 are statistically significant at 5% level in our offline evaluation.

Interestingly, the percentage of recommended items that got re-ranked to a different position among the top 5 recommendations is greater for the baseline method than for the machine learning method. In other words, the baseline approach actually effects greater re-ranking despite having less influence on the outcome. % re-ranked is 35% for the baseline and only 29% for the machine learning method.

We further take a look at the weights associated with each feature, i.e. w_1, w_2, w_3 in the inference function. After adjusting for feature variance, the ratio for $w_1:w_2:w_3 = 73:19:10$. This shows that the relevance measure from the item-anchored VUB model has the greatest influence in the personalized model. The customer price affinity feature has greater weight than the price2price feature.

* indicates statistical significance at $p < 0.05$ compared to baseline

5 CONCLUSION

In this paper, we presented our novel solution to the problem of scalable personalized recommendation systems. Our re-ranker framework isolates the effect of item relevance, learned from item anchored recommendation model, from personal preference, thus allowing us to develop personalization on top of the results from existing non-personalized models. We also opted to express customer preference (defined to target different product attributes such as brand, price, flavor, etc.) in terms of explicit signals instead of implicit representations. We introduced a couple of examples on how we can use the framework in real applications. Both case studies showed great potential of our re-ranker system through offline evaluations.

REFERENCES

- [1] Oren Barkan and Noam Koenigstein. 2016. Item2vec: neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 1–6.
- [2] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).
- [3] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikrit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1809–1818.
- [4] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S Yu. 2018. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1531–1540.
- [5] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*. Ieee, 263–272.
- [6] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
- [7] Thom Lake, Sinead A Williamson, Alexander T Hawk, Christopher C Johnson, and Benjamin P Wing. 2019. Large-scale collaborative filtering with product embeddings. *arXiv preprint arXiv:1901.04321* (2019).
- [8] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
- [9] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
- [10] Pavlos Mitsoulis-Ntompos, Meisam Hejazinia, Serena Zhang, and Travis Brady. 2019. A Simple Deep Personalized Recommendation System. *arXiv preprint arXiv:1906.11336* (2019).
- [11] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).
- [12] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.

- [13] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Yu Chen, and Chi Xu. 2017. MRLR: Multi-level Representation Learning for Personalized Ranking in Recommendation.. In *IJCAI*. 2807–2813.
- [14] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-prod2vec: Product embeddings using side-information for recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 225–232.
- [15] Mengting Wan, Di Wang, Jie Liu, Paul Bennett, and Julian McAuley. 2018. Representing and recommending shopping baskets with complementarity, compatibility and loyalty. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1133–1142.
- [16] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 968–977.
- [17] Yuexin Wu, Hanxiao Liu, and Yiming Yang. 2018. Graph Convolutional Matrix Completion for Bipartite Edge Prediction.. In *KDIR*. 49–58.