THE UNIVERSITY OF CHICAGO

CUSTOMIZED E-COMMERCE B2C RECOMMENDATIONS SYSTEM FOR AMAZON

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE SOCIAL SCIENCE DIVISION
IN CANDIDACY FOR THE DEGREE OF
MASTER OF ARTS

DEPARTMENT OF COMPUTATIONAL SOCIAL SCIENCE

BY
JINGWEN NI
12332025
Preceptor: Jon Clindaniel

CHICAGO, ILLINOIS
05/01/2023

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

This thesis is dedicated to my loving family. whose unwavering support and encouragement have been the bedrock upon which I built my academic journey. To my mentor Bingxuan Ying, preceptor Jon Clindaniel, Sabrina Nardin and peers Lu Zhang, Zhe Zhang, Haohan Shi who have helped me inspired me. and shared in the highs and lows of this incredible voyage, I am eternally grateful. To all the dreamers and seekers, may this work light the way forfuture explorers, as we continue to unlock the mysteries of our worldand beyond.

And finally. to the memories of those who have gone before useaving an indelible mark on ourlives and our hearts - this accomplishment is a testament to your enduring legacy.

With deepest gratitude Jingwan Ni

ABSTRACT

The rapid development of information technology and digitalization has promoted the spread of e-commerce. The variety of products can make customers feel overwhelmed when choosing and using them. Currently, Customized E-commerce B2C Recommendation Systems bring convenience to both customers and businesses by improving sales efficiency, solving the problem of information overload, and optimizing the supply chain. Amazon is an American multinational technology company that specializes in e-commerce and has been referred to as one of the most influential economic and cultural forces in the world. For this project, I will design a Customized E-commerce B2C Recommendation System for Amazon by analyzing Amazon Review Data (Ni, 2018), which is a collection of reviews including ratings, text, helpfulness votes, product metadata like product descriptions, category information, price, brand, image features, and links that are viewed. All analysis tools are from Amazon Web Services (AWS), such as Amazon Textract, Amazon Comprehend, Amazon A2I, Amazon Kendra, Amazon Personalize, Amazon Translate, etc. (AWS website). There are four stages in this project: retrieval, filtering, scoring, and ordering. In this paper, I will focus on the retrieval stage. The method of the retrieval stage should be based on a hybrid system called Enhanced Augmented Two-Tower Modeling, which combines Collaborative Filtering (CF) and Content-Based Filtering (Yu, Wang, Feng, Xue, 2021). The goal of the project is to build customer profiles and recommend products/services by matching customer preferences and product information based on textual information like product descriptions and customer comments.

Keywords: Customization, E-commerce, AWS, Recommendation System, Two-Tower Model, Data mining, Web usage mining.

LITERATURE REVIEW

Customized E-commerce B2C recommendation systems can improve the efficiency of sales between businesses and customers. Customization is defined as producing goods and services to meet individual customers' needs with near mass production efficiency (Jiao & Tseng, 2021), allowing customers to specify their preferences at the latest point in the supply network (Chase, Aquilano, Jacobs, 2004). E-commerce Recommendation Systems are web-based technologies that collect consumers' preferences and make recommendations (Li & Karahanna, 2015). Recommendation systems play a vital role in helping users find items of interest from a vast pool of options.

The impacts of customized recommendation systems can be generally divided into customer level and market level. For the impacts on consumers' perceptions, intentions, and decision-making when they select products, previous studies based on TAM and TRA show that recommendations will increase consumers' perceptions and intentions (Chau & Lai, 2003; Kumar & Benbasat, 2006; Liang et al., 2012) and foster positive attitudes towards the system (Chau & Ho, 2008). Customized recommendation systems also reduce the time cost and effort required when customers search for suitable products, which can be considered as transaction cost. Some studies show that the presence of customized products and the relevance of recommendations significantly affect customers' cognitions (Tam, Ho, 2012). Customized recommendation systems can also increase the number of customers' searching times and considerations (Haubl, Trifts, 2000).

Besides the customer level, customized recommendation systems also impact the market. There is a conflict in previous studies regarding sales diversity. Some scholars believe that the system limits the diversity of products and services for customers to explore (Pariser, 2011), as it only recommends homogenized products (Fleder & Hosanagar, 2009; Oestreicher-Singer & Sundararajan, 2012). In contrast, another group suggests that the system can increase heterogeneous sales by recommending products that customers ignored before. Customized

Recommendation Systems not only increase the sales of recommended products but also the sales of related products (Li & Karahanna, 2015).

However, a significant challenge in such systems is accurately scoring millions or billions of items in real-time (Yu, Wang, Feng, Xue, 2021).

From previous studies, recommendation systems are limited by product attributes and consumer preferences, either explicitly or implicitly (Li & Karahanna, 2015). As more and more unstructured consumer information becomes available online, such as reviews and social network information (Gottschlich 2013, Park 2012), there should be an opportunity to investigate a new approach to building customer profiles and matching them with textual information like product descriptions and consumer comments (Li & Karahanna, 2015). It is common to employ a two-phase approach when designing a recommender system. In this approach, a retrieval model is used to retrieve a small subset of relevant items from a large corpus based on the user's query, followed by a ranking model that ranks the retrieved items according to user clicks or ratings. To create a scalable retrieval model, many scholars first train query and item representations and then use cosine similarity between them to provide personalized recommendations for the user's query. However, in large-scale applications, the item corpus can be massive, and there may be limited user feedback data available for most queries and items. This can lead to inaccurate predictions for niche users and items. To address these challenges, the two-tower model, which utilizes deep neural network (DNN) encoders, is often employed. The "towers" refer to the two separate encoders, one for queries and the other for items, which are trained simultaneously to learn more accurate representations of both (Yu, Wang, Feng, Xue, 2021).

Although the two-tower model has many advantages, it still has some limitations. Firstly, model performance could be hindered because the item representation of the item tower needs to be pre-computed in advance for online retrieval services. This means that the forward computation of the item tower must be independent of the query tower, which results in a lack of interaction between the two towers. Secondly, the items in recommendation systems belong to diverse categories such as food, drinks, movies, etc. The number of items in each category can be severely imbalanced, with some categories having many items while others have far fewer. As a result, the model may perform poorly on categories with relatively fewer items.

To solve the aforementioned problems, I am going to implement an Enhanced Two-Tower Model. There are two main mechanisms: the Adaptive Mimic Mechanism customizes an augmented vector for each query and item as their content features, and the Category Alignment Loss aligns the representation for items from different categories during the training phase. In summary, the Enhanced Two-Tower Model has two primary benefits. First, it offers deeper insights into the information interaction of two-tower models in the retrieval task. Second, it improves the quality of item representations, particularly in scenarios where the category distribution is extremely imbalanced (Yu, Wang, Feng, Xue, 2021).

In further studies, an industrial-level recommendation system should aim for large-scale, high-volume traffic production systems that require a low-latency real-time response. The recommendation system discussed in this paper operates offline. The disadvantage of having the system running offline is the delay between user activity and recommendation generation. Thus, a near real-time recommendation system is preferred to solve the problem of delay and optimize the recommendation generation time, although the tradeoff might be to maintain a more complex system.

In addition, we should examine the homogenizing or heterogenizing effects by considering more factors, such as the accuracy of customized recommendations, to explore more impacts on product sales and customer behaviors. Moreover, we can also focus on studying the effects on related or complementary products (Li & Karahanna, 2015).

DATA

A Python script is written to scrape 2022 review data on different product categories from the Amazon website. The list of product categories is referred to in Jianmo Ni's Amazon review data (2018).

Initially, the Python script runs on a local laptop in a single thread, fetching only 50 data per second. The time it takes to retrieve a complete dataset of this size is too long. Therefore, parallel computing strategies were chosen and implemented to accelerate the process. The script is then tested on a local laptop with 4 cores and 8 threads to ensure everything is correct before launching a costly AWS EC2 Instance. The final scraper script adopts both multi-threading and multi-processing strategies to maximize efficiency and runs on a 128 vCPU AWS EC2 instance remotely. The script is able to scrape 48,000 data per second and takes around 2 hours to fetch 350 million reviews and 20 GB of compressed product metadata.

Amazon Review data is a collection of review data, metadata, and visual features. The updated version includes more reviews, newer reviews, transaction metadata like product information and product images, more detailed metadata of the product landing page such as bullet-point descriptions under the product title, technical details table (attribute-value pairs), and similar products table. Additionally, it features 5 new product categories.

In the updated 2022 version, there are 423.1 million reviews, an increase from 142.8 million in 2018. This dataset includes reviews from May 1996 to October 2022.

Review data includes: reviewerID as the ID for the reviewer, asin as the ID for the products, reviewerName as the name of the reviewer, helpful as the helpfulness rating of the review, reviewText as the text of the review, overall as the product rating, summary as the summary of the review, unixReviewTime as the time of the review (unix time), and reviewTime as the time of the review (raw).

Metadata includes descriptions, category information, price, brand, image features, and co-purchasing links, such as: asin as the ID of products, title as the name of the products, price in US dollars, imUrl as the URL of the product image, related as the related products (also bought, viewed, bought together, buy after viewing), salesRank as sales rank information, brand as the brand name, and categories as the list of categories the products belong to.

Visual features are the visual features for all products extracted from the imUrl field in metadata files (Ni 2018 & McAuley, 2014).

AWS tools will be used to store, process, and run the analyzing code. Only 10k pieces of data were taken for testing purposes and to check whether the whole recommendation system is functional or not. Since the size of the data is huge, it is impossible to have the script process the entire pack of data at once. The decision has been made to store the compressed .gz file in an AWS S3 bucket first and build an extract, transform, and load (ETL) job to move the data to AWS Redshift, a data warehouse, for further analysis.

METHOD

MODEL SELECTION

Previous studies are not sufficient for matching customer preferences and products. Hence, in this study, I will build customer profiles and recommend products/services by matching customer preferences based on products' textual information like product descriptions and customers' comments.

The initial idea for selecting a strategy for the retrieval stage was deciding between collaborative filtering and content-based filtering. However, both approaches have their own disadvantages in different scenarios. For example, collaborative filtering, in our case user-to-item recommendations, has limitations in recommending products since each individual user has their own distinct preferences, which may not align with those of any other user. In other words, there may not be any similarities or matches in tastes between different users. As for content-based filtering, also known as item-to-item recommendations, it has limitations in suggesting or introducing new interests to users beyond their existing preferences because it only makes recommendations based on users' existing interests (Pariser, 2011).

As suggested by papers, including (Yang, Yi, Chen 2020), a hybrid approach can be taken to take advantage of both collaborative filtering and content-based filtering algorithms. This approach is called the two-tower model, which is widely used in the industry. The two-tower model is an extension of Neural Collaborative Filtering. If a model has a user-side model, item-side model, and an interaction layer, it can be called a two-tower model. Neural CF uses the history of user-object interactions to construct a co-linear matrix. It uses each row of users to find similar users and then recommends items that similar users like to other users. We can divide the matrix into two parts: one is the user matrix, and the other is the item matrix. The input for this model is a one-hot vector generated by userID and a one-hot vector generated by itemID. These two vectors will be transformed into more complex vectors and produce latent vectors of users and items. The output is the dot product of the user's latent vectors and the item's latent vector. Finally, the network will be updated by comparing the output and target score and through reverse gradient propagation. In this case, the inner product is too simple to provide accurate predictions, so we can replace the dot product with Neural CF layers (Fig. 1).

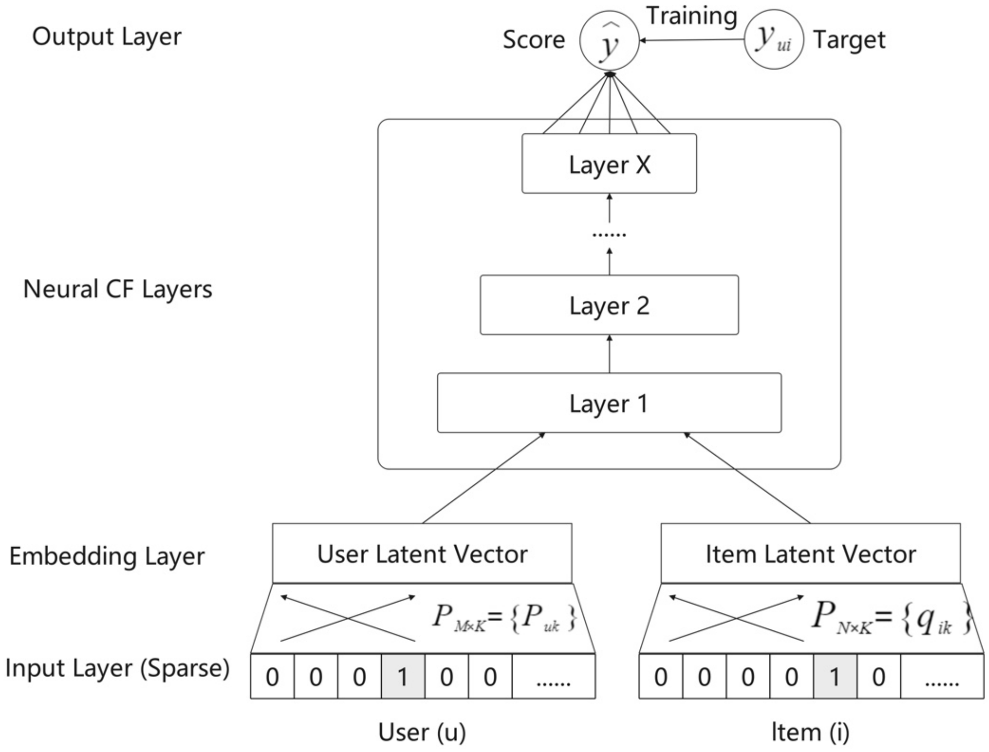

Figure 1: Neural CF in Two Tower Model

Neural CF allows a deeper interaction between user latent vectors and item latent vectors. For Neural CF, it only uses userID as an input feature for the user tower and uses itemID as an input feature for the item tower and user tower. In fact, we can add more features, like other related users and items, into the user tower.

For example(Fig. 2), YouTube adds more features to both models, such as the ID number of the video currently being watched by the user, the ID for historically watched videos, the number of times the video has been watched, and the number of times the video has been liked. L2 normalization is applied before calculating the embeddings from both towers using the dot product. The purpose of L2 normalization is to obtain augmented representations of inputs from both towers, where variables can be later selected and used in future embeddings. Dot product is then adopted to calculate the cosine similarity of vectors from the two towers.

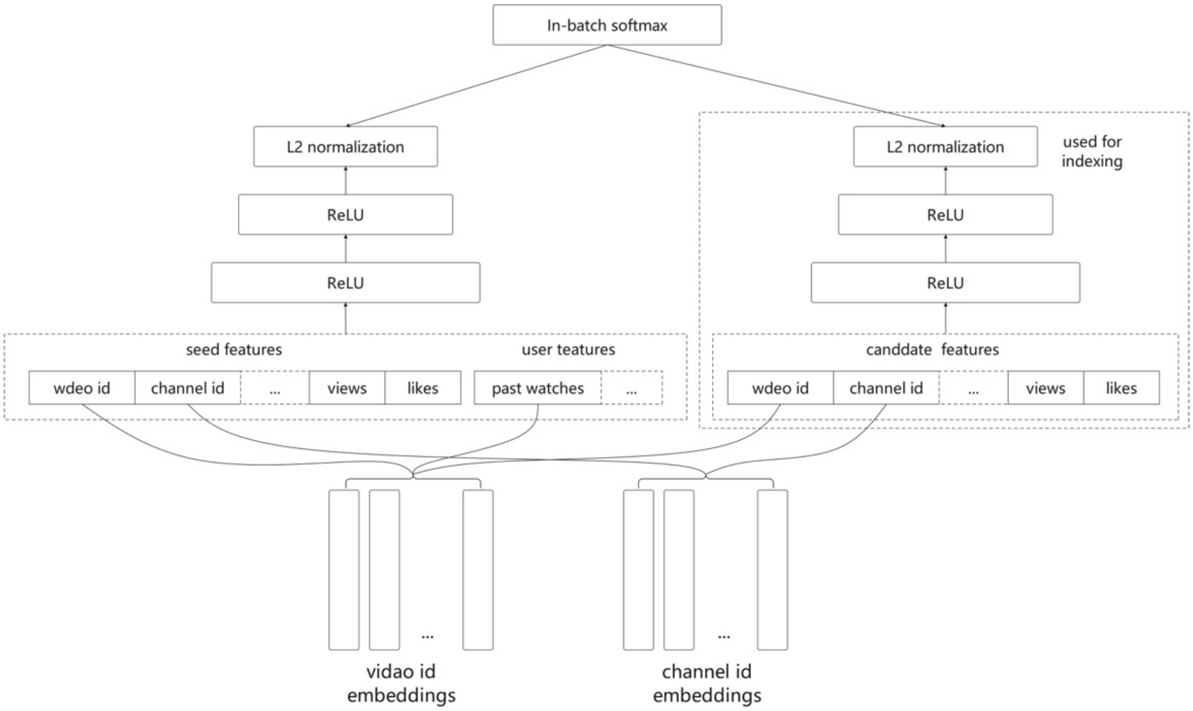

Figure 2: YouTube Two Tower Model Sample

This approach tackles the challenges presented by both algorithms, and it also solves another essential real-world problem with deep neural networks (DNNs). In industrial-scale applications, the number of product items can be enormous, while there may not be a substantial amount of feedback data available for the model to train on, as training data gathered from users' feedback is scarce or limited. The two-tower model has more advantages than EmbeddingMLP and Wide&Deep because it is much easier to be built and used.

The reason is: in the graph (Fig. 3), x and y are input features of the user and item. $u(x)$ and $v(y)$ are embedding vectors of the user and item. When applying the model, we only need to interact with $u(x)$ and $v(y)$ to get the result, so there is no need to deploy the whole model into production, and we just need to store the $u(x)$ and $v(y)$ in a database.

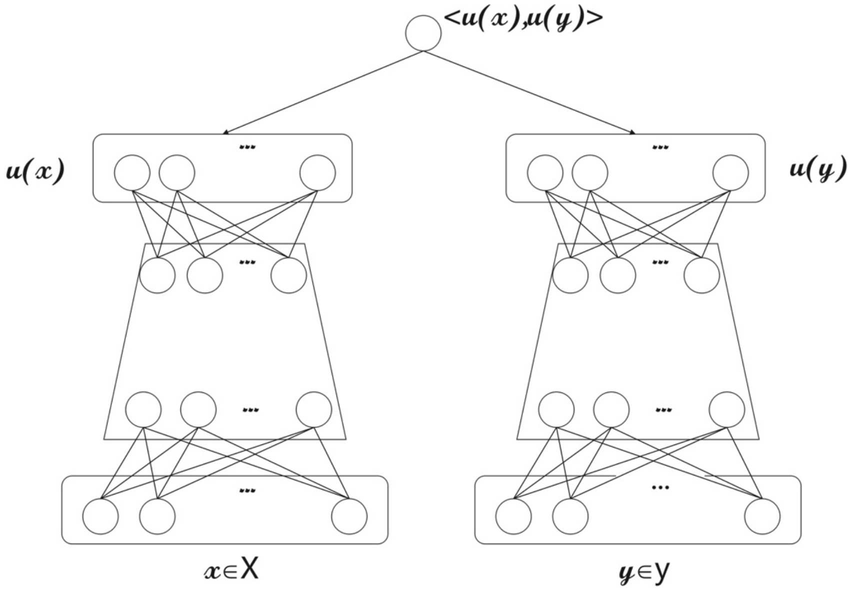

Figure 3: Two Tower Model Embedding

PROBLEMS

Despite the more precise predictions of the two-tower model, some problems have been revealed during research on the approach. The first limitation is that the model's performance and accuracy are impeded by the lack of interaction or exchange of information between the two towers. The other limitation is the imbalanced size of categories. When a category has a relatively small set of items, the model inevitably has worse performance (Yu, 2021). To address these two disadvantages, additional features were implemented to increase prediction accuracy.

SOLUTIONS

For the first limitation, “the model’s performance and accuracy are impeded by the lack of interaction or exchange of information between the two towers.” In the previous two-tower model, the interaction layer is simply made by a concatenate function with input from the item tower and user tower, so the interaction between the two towers is not adequate enough. To enhance it, I added a multi-layer neural network inside the item tower and user tower, which can enable features in towers to intersect deeply.

For the second limitation, "the imbalanced categories in size. When the category has a relatively small set of items, the model inevitably has worse performance. I added a weighted matrix w1 and a bias vector b1 before the L2 normalization.

To implement the two-tower model, the data is first separated into two groups, user reviews/ratings and product metadata, for embeddings. Embeddings are vector representations of an entity that map them to a continuous space. Each element of the vector corresponds to a feature, or a set of features associated with the entity (Alam 2020). The objective is to project users and items, or items and items, into a shared embedding space, enabling us to employ nearest neighbor-based recommendations. The following features are picked for user-to-item embedding: user_id, item_id, rating, review, summary, vote; and the list of features for item-to-item embedding is: item_id, title, price, description, also_buy, also_reviewed, sales_rank, brand, categories. Before handing the listed features to DNNs, the augmented vector from the other model is appended to the input, so it can be taken into account later and to eliminate the problems created by the independent calculation of the two towers.

DNNs help create an embedding layer for both towers. Among Feedforward neural networks, the Backpropagation algorithm, and Convolutional neural networks, Feedforward neural networks are chosen as they fit our scenario where the data process should be in one direction from input nodes to output nodes, and no repetition is needed. The number of dimensions and layers are configurable; however, to find a balance point where it doesn’t take too much time to train while it can return a relatively accurate set of recommended items and the cost on AWS resources can also be reasonable (Gu, Tandon, 2021). As a result, 8 dimensions and 16 layers are chosen, and I used Keras from TensorFlow to put the neural network into two towers and dense features in layers. L2 normalization is applied before calculating both the embeddings from both towers using the dot product. The purpose of doing L2 normalization is to obtain augmented representations of inputs from both towers where variables can be later picked and used in future embeddings. Dot product is then adopted to calculate the cosine similarity of vectors from the two towers.

I approach the retrieval problem as a binary classification task and utilize a random negative sampling method. For every query in a positive query-item pair (label $= 1$), I randomly select S items from the item collection to generate S negative query-item pairs (label $= 0$) involving the same query. I include these $S + 1$ pairs in the training dataset.

By implementing the above features, the retrieval stage can accurately return a user-defined number (in our case, 1k) of recommended items when user info and clicks are given. To avoid unnecessary recommendations, the filter stage will filter out any items that are out of stock and not age-appropriate. The scoring stage then takes filtered recommended items to do further ranking. DLRM, an open-sourced recommendation model from Meta, is adopted to produce a scored list of items from high to low, and only the top k items will be returned to users.

EVALUATION/DISCUSSION

The project is a platform for Amazon to achieve higher accuracy for recommendation system. Amazon always wants a better recommendation system by optimizing the algorithms. The goal of the system is to increase the accuracy of recommended products and lead to higher sales volume. After we input the data of customers and products, this data will be stored in Redshift and consumed by the recommendation system that we design. The recommendation system is divided into four stages: retrieval stage, filtering stage, scoring stage, and ordering. The final returns should be the scores, which are the criteria for recommendations and then store scores in DynamoDB. In this paper, I only focus on retrieval stage by improving two-tower model performance in two aspects. First, increasing the interaction between two towers to. Second, providing a solution to deal with imbalanced data which can make this model fits the real-world business.

The recommendation system has reproducibility if scholars follow the instructions in this paper. The details of data processing and ranking algorithms will be specified later.

FURTHER INVESTIGATION

To develop an industrial-level recommendation system that caters to large-scale high-traffic production systems and provides low-latency real-time response, it is important to address the current system's offline operation, which leads to delays in generating recommendations. To overcome this issue, a near real-time recommendation system is necessary, although it may require a more complex setup. Additionally, to fully understand the impact of the recommendation system on product sales and customer behavior, it is crucial to consider factors such as the accuracy of personalized recommendations and the effects on related or complementary products. This would help to determine if the recommendation system homogenizes or heterogenizes the customer's purchasing behavior (Li & Karahanna, 2015).

BIBLIOGRAPHY

Li, Seth Siyuan and Karahanna, Elena (2015) "Online Recommendation Systems in a B2C E-Commerce Context: A Review and Future Directions," Journal of the Association for Information Systems, 16(2), . DOI: 10.17705/1jais.00389 Available at: https://aisel.aisnet.org/jais/vol16/iss2/2

Lopes, P., & Roy, B. (2015, March 25). Dynamic recommendation system using web usage mining for e-commerce users. Procedia Computer Science. Retrieved November 4, 2022, from https://www.sciencedirect.com/science/article/pii/S1877050915003221

Zhang, Y., & Jiao, J. (R. (2006, June 5). An associative classification-based recommendation system for personalization in B2C e-commerce applications. Expert Systems with Applications. Retrieved November 4, 2022, from https://www.sciencedirect.com/science/article/pii/S0957417406001515?casa_token=WP0EWiyyNEYAAAAA%3AtfPmCFY8sDycQ4CFhe4op9XXxCz6x3qJjUHtWRuzy0LA3JVyHZdlpB4qpT8sz1mwB0XnTr6i

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl fsarwar, karypis, konstan, riedlg@cs.umn.edu GroupLens Research Group / Army HPC Research Center Department of Computer Science and Engineering University of Minnesota Minneapolis, MN 55455

Technology, W. U. of. (n.d.). Campus life. Wuhan University of Technology. Retrieved November 4, 2022, from http://english.whut.edu.cn/

Guo, Y., Wang, M., & Li, X. (2017, March 13). Application of an improved apriori algorithm in a mobile e-commerce recommendation system. Industrial Management & Data Systems. Retrieved November 4, 2022, from https://www.emerald.com/insight/content/doi/10.1108/IMDS-03-2016-0094/full/html

Gottschlich, J., Heimbach, I., & Heimbach, I. (2013). The value of users' Facebook profile data generating product recommendations for online social shopping sites. Paper presented at the European Conference on Information Systems.

Jiao, J., & Tseng, M. (2001). Mass customization handbook of industrial engineering B2handbook of industrial engineering (Vol. 3). New York: Wiley.

Park, S.-H., Huh, S.-Y., Oh, W., & Han, S. P. (2012). A social network-based inference model for validating customer profile Data. MIS Quarterly, 36(4), 1217-1237.

Chau, P. Y. K., & Ho, C. K. Y. (2008). Developing consumer-based service brand equity via the Internet: The role of personalization and trialability. Journal of Organizational Computing and Electronic Commerce, 18(3), 197-223.

Chau, P. Y. K., & Lai, V. S. K. (2003). An empirical investigation of the determinants of user acceptance of Internet banking. Journal of Organizational Computing & Electronic Commerce, 13(2), 123-145.

Kumar, N., & Benbasat, I. (2006). The influence of recommendations and consumer reviews on evaluations of websites. Information Systems Research, 17(4), 425-439.

Kumar, R. L., Smith, M. A., & Bannerjee, S. (2004). User interface features influencing overall ease of use and personalization. Information & Management, 41(3), 289-302.

Häubl, G., & Murray, K. B. (2003). Preference construction and persistence in digital marketplaces: The role of electronic recommendation agents. Journal of Consumer Psychology, 13(1/2), 75-91.

Häubl, G., & Trifts, V. (2000). Consumer decision making in online shopping environments: The effects of interactive decision aids. Marketing Science, 19(1), 4-21.

Pariser, E. (2011). Beware online filter bubbles. Retrieved from http://www.ted.com/talks/eli_pariser_beware_online_filter_bubbles.html

Park, S.-H., Huh, S.-Y., Oh, W., & Han, S. P. (2012). A social network-based inference model for validating customer profile Data. MIS Quarterly, 36(4), 1217-1237.

Hassanein, K., & Head, M. (2005). The impact of infusing social presence in the Web interface: An investigation across product types. International Journal of Electronic Commerce, 10(2), 31-55.

Fleder, D. M., & Hosanagar, K. (2009). Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. Management Science, 55(5), 697-712.

Oestreicher-Singer, G., & Sundararajan, A. (2012a). Recommendation networks and the long tail of electronic commerce. MIS Quarterly, 36(1), 65-A64.

Oestreicher-Singer, G., & Sundararajan, A. (2012b). The visible hand? Demand effects of recommendation networks in electronic markets. Management Science, 58(11), 1963-1981.

Schiaffino, S., & Amandi, A. (2004). User-interface agent interaction: Personalization issues. International Journal of Human-Computer Studies, 60(1), 129-148.

Justifying recommendations using distantly-labeled reviews and fined-grained aspects Jianmo Ni, Jiacheng Li, Julian McAuley Empirical Methods in Natural Language Processing (EMNLP), 2019

Yantao Yu, Weipeng Wang, Zhoutian Feng, Daiyue Xue. 2021. A Dual Augmented Two-tower Model for Online Large-scale Recommendation. In Proceedings of DLP-KDD 2021. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

Yang, Ji, et al. "Mixed negative sampling for learning two-tower neural networks in recommendations." Companion Proceedings of the Web Conference 2020. 2020.

Zhao, Chi, and Ivan Blekanov. "Two Towers Collaborative Filtering Algorithm for Movie Recommendation."   8.1 (2021): 397-401.

Gu, W., Tandon, A., Ahn, YY. et al. Principled approach to the selection of the embedding dimension of networks. Nat Commun 12, 3772 (2021). https://doi.org/10.1038/s41467-021-23795-5

https://www.jianshu.com/p/6aff15ec4b80