Deep neural network marketplace recommenders in online experiments

Simen Eide
Schibsted Media Group
Oslo, Norway
simen.eide@schibsted.com

Ning Zhou
Schibsted Media Group
Oslo, Norway
ning.zhou@schibsted.com

ABSTRACT

Recommendations are broadly used in marketplaces to match users with items relevant to their interests and needs. To understand user intent and tailor recommendations to their needs, we use deep learning to explore various heterogeneous data available in marketplaces. This paper focuses on the challenge of measuring recommender performance and summarizes the online experiment results with several promising types of deep neural network recommenders - hybrid item representation models combining features from user engagement and content, sequence-based models, and multi-armed bandit models that optimize user engagement by re-ranking proposals from multiple submodels. The recommenders are currently running in production at the leading Norwegian marketplace *FINN.no* and serves over one million visitors everyday.

CCS CONCEPTS

• **Information systems** → **Content ranking**; *Personalization*;

KEYWORDS

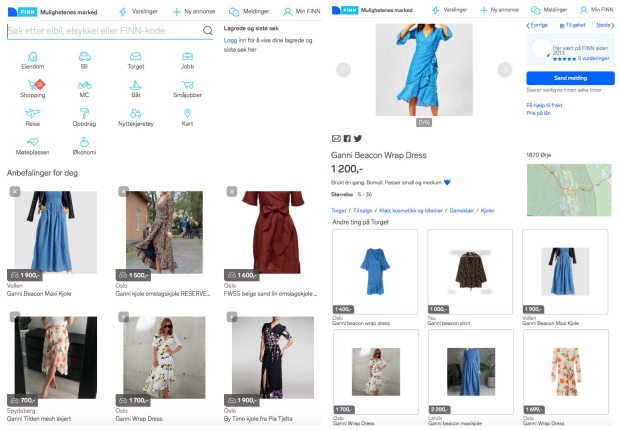
recommendation system; deep learning; marketplace

ACM Reference Format:
Simen Eide and Ning Zhou. 2018. Deep neural network marketplace recommenders in online experiments. In *Twelfth ACM Conference on Recommender Systems (RecSys '18), October 2–7, 2018, Vancouver, BC, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3240323.3240387

1 INTRODUCTION

Marketplaces are platforms where users buy and sell various types of items. The items can range from low-value ones such as books and clothes to high-value ones such as cars and real estate properties. Sellers can also post non-tangible items such as job openings and services. Many marketplace sellers are non-professional individuals selling used items, therefore marketplaces can be viewed as a special type of e-commerce that involves unique items across multiple categories from a very large and fragmented seller group.

Recommendation systems are broadly used in marketplaces to match buyers with items relevant to their interests and needs. It is more challenging than the standard e-commerce product recommendation due to the following reasons: (1) Marketplace items are often secondhand and therefore in a sense unique. This results in a short lifespan for a specific item. Thus cold-start issues are very prominent. (2) The items are often poorly described by unstructured data, making conventional cold-start algorithms harder to apply. (3) The interaction between buyers and sellers is often tricky to track, as it can happen outside the platform. Transactions cannot always be confirmed, so the recommender system must be able to change focus from sold items quickly to remain relevant.



(a) Item feed **(b) Similar item**

Figure 1: Sample marketplace recommendations.

There is already substantial research on recommender systems validated on offline metrics. In this paper, we describe three new marketplace recommenders and benchmark them through online experiments against industry standard models such as matrix factorization to test their performance in a production environment where short lifespan and low quality content are common features of the items recommended. The three models we describe here are: (1) A hybrid item-item recommender that utilizes behavior data, image and text to build a robust item representation against cold-start issues. (2) A sequence based user-item recommender that is time-aware and can utilize the hybrid item representation above to quickly build a user's profile. (3) A higher-level multi-armed bandit algorithm that prioritizes between multiple submodel recommendations into a personalized item feed. It allows the feed to cover both long and short term user interests.

The production environment for testing is the leading Norwegian marketplace *FINN.no*. It has over one million active items in 10+ categories and 200+ subcategories for sell, and serves over one

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 the author(s) 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.
RecSys '18, October 2–7, 2018, Vancouver, BC, Canada
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-5901-6/18/10…$15.00
https://doi.org/10.1145/3240323.3240387

million visitors per day. There are two recommendation features at *FINN.no*: an item feed on the frontpage and a similar item recommendation widget on the item detail page. One sample of each is shown in Figure 1. The hybrid model is used for the similar item recommendation, while the rest two are used for the item feed.

2 RELATED WORKS

Search and discovery are the two main ways for users to find relevant items on a content platform. Using recommendations to improve the discovery experience has been a hot topic in recent years. Both collaborative filtering [1] [18] and content based methods [3] [9] are commonly used in item similarity ranking for e-commerce. Recent works like LightFM [14] combine the two to address the cold-start problem in recommendations. The state-of-the-art recommenders such as [21] and [15] often use learning-to-rank to model from a complex set of features including text, image, user profile, behavior, etc. This strategy is effective with marketplaces recommenders as well [8]. Moreover, models of cascade [16] or sequential attention [22] that consider a longer user behavior history show particularly good results. Our sequential recommender draws inspiration from [2] and shows positive performance improvements in online experiments. Multi-armed bandits for prioritizing among multiple sources is applied in different use cases. For example, Pinterest [7] uses a blending algorithm to generate a mixed feed of their recommendation content and social content. We compared several multi-armed bandit models to leverage the behavior-content complementation in the marketplace scenario.

Recommenders are usually evaluated through click and conversion rates from online controlled experiments [12]. There are offline metrics such as predicted ratings and ranking correlation [10] that can be used with lower preparation cost, and serve as a proxy to the online tests.

3 EXPERIMENTATION PLATFORM

A robust experimentation platform for production requires both a fast iteration cycle from idea to production and a risk-averse test procedure to secure the product quality. From the business perspective, the goal of a recommendation system is to increase the amount of conversions, so we choose to rely on A/B tests to select good candidate models based on conversion metrics in production.

Our experimentation system consists of offline and online experiments. We use the offline ones mainly to quickly rule out the candidate models that are clearly underperforming and therefore not feasible for online testing. This allows us not only to utilize online experimentation resources more efficiently, but also to lower the risk of poor user experience. We use the offline metric Hit Rate@n (*HR@n*) [6] as a proxy for online conversion metrics. Historical data are splitted into a training set $D_{t<t_{th}}$ and a test set $D_{t\geq t_{th}}$ by a specific date t_{th}. A model is trained on $D^u_{t<t_{th}}$ and recommends top n items for each user u. *HR@n* is defined as the average per user hits of the top n items found in $D^u_{t\geq t_{th}}$.

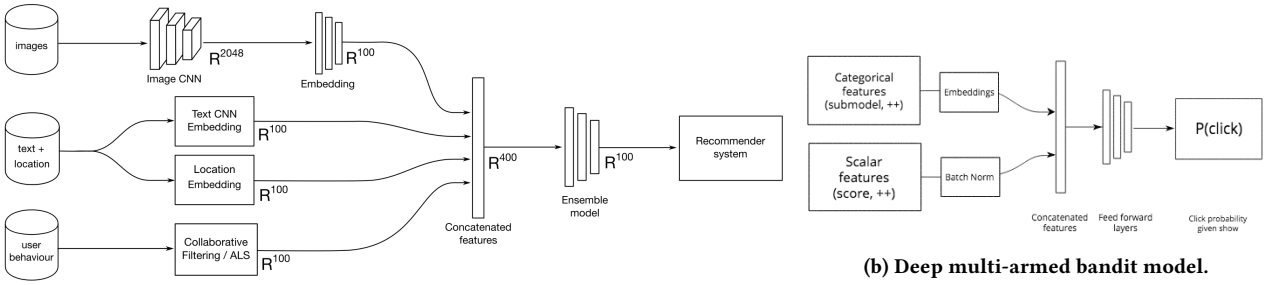
The candidates that perform better or in the same range as the baseline model in offline tests are deployed online to A/B test in production with gradually higher traffic. We generally follow [13] for the experiment procedure design. Online experiments are evaluated with click through rate and other conversion metrics such as seller contacting rate. We monitor all experiments for significance using a binomial test. During an A/B test, we also consider the stability of the result over time. In addition to aggregating all the samples to calculate the final test result, we monitor the model performance in smaller time bins during the whole test. If model B is not better than model A, the probability of model B performing better than model A in almost all time bins will be very small. When model B shows improvement but not significant, we gradually increase its traffic up to 50% in a longer A/B test to see if it will reach significance.

4 HYBRID ITEM REPRESENTATION MODEL

Similar to Lightfm[14], we combine content-based features with user behaviors to solve the cold start challenge of collaborative filtering. We design a hybrid model mixing item representations from user behavior, text, image and location to find similar items. As shown in Figure 2a, we first train the item representations independently and then ensemble them using a separated model.

The individual features are generated in the following way: (i) The behavior-based item representation is from matrix factorization implemented with the industry standard Alternating Least Squares (ALS) [19]. The training data consists of 20-day look-back of item clicks and conversion signals such as contacting seller. The conversion signals are stronger and therefore acquired more weights. All hyper-parameters are tuned through offline and online experiments. (ii) The textual features are from the unstructured free text in the item title and description. We first train an item category classifier to map the text to the seller-assigned category, and then extract the top layer of this classifier as the textual item representation. The classifier is a variant of the Convolutional Neural Network (CNN) architecture of [4] using a word2vec [17] model pre-trained on marketplace corpus. (iii) The image features are generated by training a model to predict the item title from its image. The title is projected into word embeddings by the word2vec model mentioned above. We use the title instead of the category to classify the images. This gives us a richer feature space. For example, it enables the item representation to distinguish between "wedding dress" and "summer dress", despite both belong to the same category "dresses". The model uses the penultimate layer of a pre-trained *Inception-v3* model [20] and stacks seven linear feed-forward layers on top to predict the title in the word embedding space. It is then trained by minimizing the mean squared error between the predicted title embeddings and the real title embeddings. (iv) We do not use the simple geographical distance to represent location, because it can be misleading. There are hidden factors such as population density and ease of transport that affect the impact of location. We train the location representation based on the historical user behavior, since the items a user showed interest in implicitly tell us a lot about the hidden factors. Similar to (i), we factorize a user-postcode matrix and use the postcode embedding as the location representation.

The different item representations are merged into a hybrid one using a Siamese network. All the item representations are concatenated and passed through an attention layer. The attention layer allows the model to focus more on textual and image features if the collaborative filtering features are missing, and vice versa. Then, a towering feed-forward network compresses the features into a 100-dimensional item representation due to the capacity limit



(a) Hybrid item representation model.

(b) Deep multi-armed bandit model.

Figure 2: Overview of the recommender models.

of our serving infrastructure. Two items are compared using the cosine similarity of this representation. The training data consists of co-converted item pairs, i.e. two items that get conversions from the same user on the same day, and negative sampling item pairs that are unlikely to co-convert. The underlying assumption is that co-converted items are likely to be similar.

5 SEQUENCE-BASED MODELS

Sequence-based models look into a user's click history and predict what items they will click next. In contrast to matrix factorization, such models are time-aware and can take recent clicks into account more in the predictions. They take the n most recent items clicked by a user and project them into the item space defined by the hybrid item representation described in Section 4. The item sequence is fed through a recurrent neural network. At every step t, the models use the click history sequence $\{x_{t-n}, ..., x_t\}$ to predict a sequence $\{\hat{x}_{t+1}, ..., \hat{x}_{t+k}\}$ of the future k steps. The accuracy is calculated by the cosine similarity of the predicted sequence $\{\hat{x}_{t+1}, ..., \hat{x}_{t+k}\}$ at t and the actual clicked sequence $\{x_{t+1}, ..., x_{t+k}\}$ at $t + k$.

In [2], two-layer GRU was shown to be the optimal architecture. We also tested different variants using LSTM or GRU, adding additional stacked recurrent layers, or adding attention to the outputs. However, we did not observe any additional significant improvements over using one straight forward GRU layer.

6 MULTI-ARMED BANDIT MODELS

The multi-armed bandit models can generate a live item feed for a specific impression. The bandits are not recommenders by themselves but re-rankers that receive proposals from independent submodels as input and re-rank all the proposed items from most to least relevant by estimating their click probabilities with a value function. A submodel can be a recommender of any type, e.g. sequence-based or matrix factorization, that returns its top proposals along with their corresponding scores. Typically, a bandit is connected to 6-10 submodels. In order to avoid local minimal during training, we adopt a simple epsilon-greedy policy and add 5% random items in every recommended list to the bandit.

This approach differs from the ones such as [5] that use simple models as the first filter into a more complex model. Instead, the bandit enables us to directly leverage the results from well-performing models tuned for different scenarios. It utilizes the scores from each submodel and focuses on evaluating submodel reliability as well as contextual information such as visiting time, device, landing page type (main frontpage or categorical frontpage), etc.

The row-separated feed described in Section 6.1 is just a naive baseline for experimentation purpose. We propose two different value functions to estimate click probabilities: *regression bandit* in Section 6.2 and *deep classification bandit* in Section 6.3. The former is selected mainly due to its good interpretability, whereas the latter allows us to increase value function complexity and get a better online performance.

6.1 Row-separated feed

This simple baseline has a fixed number of rows and serves one submodel per row. The row order is also fixed based on the individual performance of each submodel. We choose this setup because it is commonly used by many e-commerce recommendation plug-ins.

6.2 Regression bandit

The regression bandit, as its name, using regression to approximates the click probability based on submodel score, submodel type and contextual information. Submodel scores are binned into 10 buckets. The remaining features are categorical and one-hot encoded. The recommendation impressions are grouped across these features and the target value is the average number of clicks per group. A ridge regression model is fit on the encoded features, as the weights of the unregularized versions often explode.

6.3 Deep classification bandit

The deep classification bandit estimates the click probability as a classification problem. This allows us to use more complex functions taking in a larger set of features than the regression bandit.

The model input is a mix of scalars (submodel score, item position, hour of day, etc.) and categorical variables (submodel type, location, device, weekday, etc.). The scalars are normalized through batch normalization, and the categorical variables are one-hot encoded. A towering feed-forward network is applied to the features and finally outputs a click probability. The dataset contains an unequal ratio of click v.s. view events (i.e. only a small ratio of recommended items were clicked) that can cause severe instabilities in training. This is solved by introducing class weights in the loss function and applying L2 regularization on the model weights.

7 RESULTS AND DISCUSSIONS

We conducted online experiments on both the similar item widget and the item feed shown in Figure 1. Similar item widget is shown at the bottom or the sidebar of an item detail page, assuming that users view the item details because they are interested in this item and therefore interested in similar items. The item feed is usually displayed at the main and categorical frontpages of the marketplace, aiming to provide new and relevant items for users. We choose to test the hybrid item representation in the similar item widget while the multi-armed bandits and sequence-based model in the item feed, because the item-item type of models are not capable to create the serendipity experience desired in the item feed scenario [11].

The experiment results are summarized in Table 1. The absolute click thorough rate (CTR) varies a lot due to the seasonal effect, new feature launches, etc. Hence we report the more stable CTR improvement $\Delta CTR = (CTR_B - CTR_A)/CTR_A$ when comparing experiments from different periods. By default, the A/B tests last for one week to avoid seasonal impact of weekends and accumulate around one million impressions of the similar item widget and 5-10 million impressions of the item feed. For simplicity, we only show the results of the best of each type after hyper-parameter tuning.

In the similar item scenario, the pure content-based item representation model is 7.7% worse than the matrix factorization baseline, but the hybrid model outperformed the baseline by 23.8%. This indicates that the user behavior signals give more relevant recommendations than content-based features but still suffer a lot from cold start. Throughout the experiments, we observed many examples where the hybrid model used content-based features on items with very few clicks, whereas it gave similar results to the matrix factorization approach for items with abundant clicks.

Sequence-based models generally outperformed matrix factorization in the online experiments. When $n = 1$, they collapse into an item-item recommender making predictions only using the last seen item. We require $n > 1$ in order for the models to generalize and find out that a $n = 15$ look-back with a $k = 5$ prediction horizon gives the best performance of 21.2% CTR improvement.

The multi-armed bandit models were tested in two groups with different baselines, because we introduced a major user interface (UI) redesign in between. When we tested the regression bandit against the row-separated feed baseline, we observed 50.1% CTR improvement. We attribute the improvement mainly to that the regression bandit can mix submodels across the rows and not constrained by the fixed order. An interesting observation is that the scores from matrix factorization submodels do not have a monotonous increasing relationship with CTR. In our experiments, CTR keeps increasing before scores reach around 0.8, but then falls quickly after that. Investigations show that these ultra high score items have some viral tendencies and do not reflect personal taste, therefore do not transform into clicks. We introduce a break point in the regression to allow the click probability estimation to fall after a threshold, and this gained another 5% CTR improvement. The bandit also allowed us to add more submodels to the feed and opened up the UI flexibility to increase the number of items recommended and submodels connected. After redesigning the UI, CTR increased even further. The two main drawbacks of the (linear) regression bandit are that it cannot estimate the observed non-linear relationship between submodel scores and CTR, and it is not personalized. The regression function cannot handle the variable dimension explosion by introducing per user aggregation.

Table 1: Experiment Results.

Model Type	Model A	Model B	Δ CTR
item-item	matrix factorization	content-based	-7.7%
item-item	matrix factorization	hybrid	23.8%
user-item	matrix factorization	sequence-based	21.2%
multi-armed	row-separated feed	regression bandit	50.1%
multi-armed	regression bandit	deep bandit	10.0%

We observed around 10% CTR improvement from testing the deep classification bandit against the regression bandit. The deep model is personalized, for it is capable to use more features including user embeddings.

During the experiments, we also tried several promising approaches but did not succeed: (1) We tried factorization machines to solve the cold start problem, but the engineering cost was quite high and did not show significant CTR improvement, so we ended up focusing on hybrid models instead. (2) We tried to train models with only strong signals such as repeating visits and messaging, but those models turned out to perform worse due to the low data volume in a short look-back time. (3) Though both previous works and user studies show that diversity matters a lot for the item feed, when we tried to optimize explicitly with a diversity metric based on category count, we did not get significant improvements. (4) Both the regression and the deep classification value functions had large instabilities during training, and a lot of care had to be done to stabilize it for production usage.

8 CONCLUSIONS

In this paper, we propose three new marketplace recommenders - hybrid item representation, sequence-based model, multi-armed bandit, and analyze their performance in online experiments from a production setting. The results demonstrate the effectiveness of combining collaborative filtering and content features for better item representation in cold start and in sequence-based models. We also present a successful use case of bandits in recommendations as a high-level re-ranker on top of other recommenders. These bandits are useful to utilize contextual information and to combine multiple domain-specific recommenders.

For future works, there are still a lot to explore in the above-mentioned models. In general, we have not explored the content features sufficiently. Adding more content features into the item representations should reduce cold-start problems on user or item more effectively. The exploration strategies of the bandit models can be further improved: New submodels still take a long time to catch up. Moreover, clicks and transactions in marketplaces do not happen instantaneously. The transaction of an item often takes place hours or days after an user first views the item, so the "reward" of recommendations can arrive with delay. Defining the models under a reinforcement learning framework may give them an incentive to explore more broadly and provide a better user experience.

ACKNOWLEDGMENTS

This project is a team effort from the recommendation team at *FINN.no*. The authors would like to thank Audun M. Øygard, Thorkild Stray, Bjørn Rustad and Nicola Barbieri for their contribution to this paper. We also thank the anonymous reviewers for their helpful comments.

REFERENCES

[1] Oren Barkan and Noam Koenigstein. 2016. Item2vec: neural item embedding for collaborative filtering. In *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*. IEEE, 1–6.

[2] Cedric De Boom, Rohan Agrawal, Samantha Hansen, Esh Kumar, Romain Yon, Ching-Wei Chen, Thomas Demeester, and Bart Dhoedt. 2017. Large-Scale User Modeling with Recurrent Neural Networks for Music Discovery on Multiple Time Scales. *CoRR* abs/1708.06520 (2017).

[3] Konstantinos Christidis and Gregoris Mentzas. 2012. A Topic-Based Recommender System for Electronic Marketplace Platforms. *2012 IEEE 24th International Conference on Tools with Artificial Intelligence* 1 (2012), 381–388.

[4] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12, Aug (2011), 2493–2537.

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. 191–198.

[6] Mukund Deshpande and George Karypis. 2004. Item-based top-N Recommendation Algorithms. *ACM Trans. Inf. Syst.* 22, 1 (Jan. 2004), 143–177.

[7] Stephanie deWet. 2017. Personalized content blending in the Pinterest homefeed. 2017 Net ix Workshop on Personalization, Recommendation and Search.

[8] Simen Eide, Audun M. Øygard, and Ning Zhou. 2018. Five lessons from building a deep neural network recommender for marketplaces. ACM KDD'18 Deep Learning Day.

[9] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit 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.

[10] Asela Gunawardana and Guy Shani. 2009. A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *J. Mach. Learn. Res.* 10 (Dec. 2009), 2935–2962.

[11] Marius Kaminskas and Derek Bridge. 2016. Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems. *ACM Trans. Interact. Intell. Syst.* 7, 1, Article 2 (Dec. 2016), 2:1–2:42 pages.

[12] Ron Kohavi, Alex Deng, Brian Frasca, Toby Walker, Ya Xu, and Nils Pohlmann. 2013. Online Controlled Experiments at Large Scale. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*. 1168–1176.

[13] Ron Kohavi, Alex Deng, Roger Longbotham, and Ya Xu. 2014. Seven Rules of Thumb for Web Site Experimenters. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. ACM, New York, NY, USA, 1857–1866. https://doi.org/10.1145/2623330.2623341

[14] Maciej Kula. 2015. Metadata Embeddings for User and Item Cold-start Recommendations. In *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015.*, Vol. 1448. 14–21.

[15] David C. Liu, Stephanie Rogers, Raymond Shiau, Dmitry Kislyuk, Kevin C. Ma, Zhigang Zhong, Jenny Liu, and Yushi Jing. 2017. Related Pins at Pinterest: The Evolution of a Real-World Recommender System. In *Proceedings of the 26th International Conference on World Wide Web Companion*. 583–592.

[16] Shichen Liu, Fei Xiao, Wenwu Ou, and Luo Si. 2017. Cascade Ranking for Operational E-commerce Search. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1557–1565.

[17] 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.

[18] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. ACM, 285–295.

[19] Apache Spark. 2018. Apache Spark MLlib - Collaborative Filtering. https://spark.apache.org/docs/2.1.0/mllib-collaborative-filtering.html.

[20] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2818–2826.

[21] 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 (RecSys '16)*. 225–232.

[22] C. Zhou, J. Bai, J. Song, X. Liu, Z. Zhao, X. Chen, and J. Gao. 2018. ATRank: An Attention-Based User Behavior Modeling Framework for Recommendation. In *Proceedings of the thirty-second AAAI Conference on Artificial Intelligence*.