Deep Inventory *Time Translation* to Improve Recommendations for Real-World Retail

Bobby Prévost, Jonathan Laflamme Janssen, Jaime R. Camacaro & Carolina Bessega
{bobbyp,jonathanl,jaimec,carolinab}@stradigi.ai
Stradigi AI
Montréal, Québec, Canada

ABSTRACT

Recommender systems are an important component in the retail industry, but the constantly renewed inventory of many companies makes it difficult to aggregate enough data to fully harness the benefits of such systems. In this paper, we describe a technique that significantly improves the accuracy of the recommendations, validated on a real store transaction history, by performing a *time translation* that maps out-of-stock items to similar items that are currently in stock using deep features of the products. This greatly reduces the dimension of the item–item interactions matrix while preserving all the dataset entries, which mitigates the sparsity of the dataset, and provides an original solution to the cold-start problem. We also improve the coverage at no accuracy cost by favouring less popular items within a small radius in the feature space while applying the *time translation* mapping. Finally, by modelling item–item rather that user–item correlations, we are able to update the recommendations for a given user in real-time, without re-training, as the user's history receives new entries.

KEYWORDS

Recommender Systems; Cold-Start Problem; Personalization; Retail; Deep Learning; Convolution Neural Network

ACM Reference format:
Bobby Prévost, Jonathan Laflamme Janssen, Jaime R. Camacaro & Carolina Bessega. 2018. Deep Inventory *Time Translation* to Improve Recommendations for Real-World Retail. In *Proceedings of Twelfth ACM Conference on Recommender Systems, Vancouver, BC, Canada, October 2–7, 2018 (RecSys '18),* 5 pages.
https://doi.org/10.1145/3240323.3240380

1 INTRODUCTION

In the context of increasing competitiveness in online retail, it is crucial for a seller to refine its user experience, a key aspect of which are recommendations. Industry and research leaders [7, 18, 20] are relying on recommender systems and the vast amount of data at their disposal to produce relevant suggestions which, in turn, provide an enhanced and personalized customer experience, and drive sales. This reality applies to all retailers, no matter their sizes. It is, however, significantly more challenging for a smaller agent to acquire a meaningful amount of data, i.e. where clear correlations can be extracted and used for recommendations [13]. This problem is exacerbated in retail industries with higher inventory rotation rates, which then also systematically face the cold-start problem for new items.

Many retailers are indeed carrying only a few hundreds of products, which are regularly replaced by new models. In such situations, they are confronted with a massive amount of historical transactions with products that are no longer available and possess no connections with products in stock. The sparsity of the user–item interactions — naturally high — is bound to persist as the amount of items is constantly growing, but the number of available products remains constant. This important aspect and limitation of retail is seldom considered in the literature [9].

The objective of this paper is to elaborate a model where recommendations can be produced for individual users taking into account a constantly renewed inventory, while minimizing the amount of lost transaction data and the quantity of business rules needed to obtain a high coverage of cold items.

First, to reduce sparsity, we elaborated a *time translation* scheme to map discontinued items to the most similar ones currently in stock by using pictures of the products and convolutional neural networks (CNN). Then, we increased the flexibility of the *time translation* algorithm by allowing it to associate a discontinued item to the least popular in-stock item within a given similarity tolerance, thus reaching a higher coverage in the recommendations at no accuracy cost. Since the supervised learning of item–item implicit interactions necessitates negative samples, we draw them from a popularity distribution that maximizes the products covered by the recommendations. This preserves the item–item correlations present in the dataset instead of favouring single item popularity [10]. Finally, to learn the item–item implicit interactions and generalize from them, which further increases coverage, we used a deep learning based recommender system [21] — the Wide & Deep model [6].

2 PROPOSED APPROACH

The dataset used for this work has been privately provided by a company specializing in the production and sale of clothing. The subset used contains 110 000 transactions among 5700 anonymised users and 4900 products. Out of these products, only 400 are in stock — 69 000 transactions were conducted with item currently out of stock. The dataset contains transactions conducted both online and in-store. Each product is associated with a label containing its category (e.g. blouse, dress, blazer) and several pictures used to show the product on the company's website.

RecSys '18, October 2–7, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5901-6/18/10…$15.00
https://doi.org/10.1145/3240323.3240380

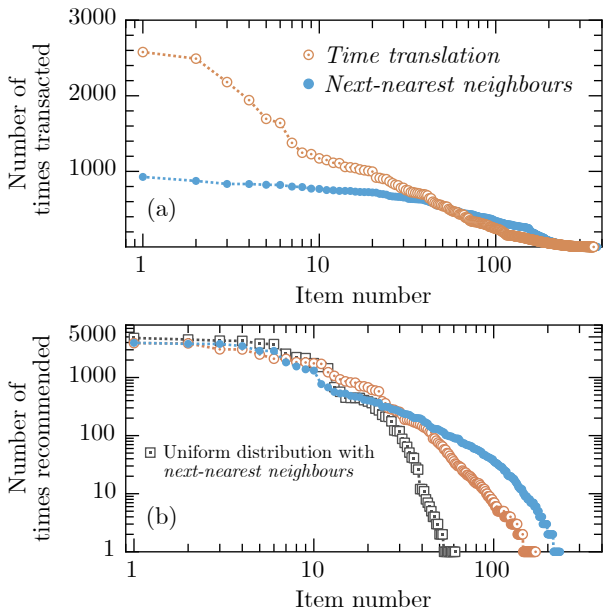

Figure 1: Figure (a) shows the item popularity after applying the *time translation*. The blue circles represent the distribution when *next-nearest neighbour* selection is used, whereas the open orange circles represent the distribution obtained without it. Figure (b) shows the popularity of each item in the top-8 recommendations for all users in the dataset. We also show the decreased coverage obtained when the negative samples are drawn from a uniform distribution.

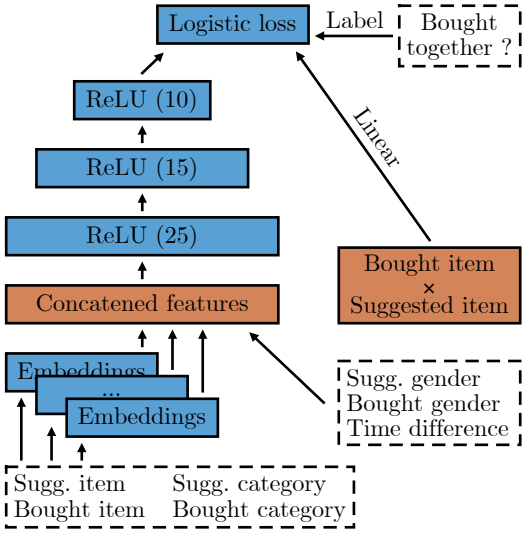

Figure 2: Architecture of the Wide & Deep network [6] used for predicting the probability that a given pair of products were bought together. The dashed boxes contain the different features used. The categorical features are individually embedded and concatenated with the numerical features before being fed into the network. Rectifier linear unit (ReLU) are used for all layers except the last, where a sigmoid is used. The *wide* (linear) part is fed the *cross-product* transformation [6] of suggested and bought products.

2.1 *Time translation*

Whenever products go out of stock, all the transactions conducted with them become unusable for generating recommendations to new users as they cannot interact with these products, which exacerbates the sparsity problem. To address this, we applied our *time translation* scheme to these products, which individually maps them to an in-stock product. The procedure is as follows:

- All images are sequentially encoded using an ensemble of CNNs [11, 14, 19]. Each CNN was pre-trained on ImageNet [8] and the weight of each network in the ensemble was optimized to maximally distinguish different textures, colours, and styles.
- Features are extracted in the bottleneck layer — before the softmax classification layer — and a distance is calculated between all products using an approximate nearest neighbours algorithm (NN) [4, 16].
- Each product out of stock is associated to the closest product in stock within the same category in the feature space.

A distance threshold $d_{\max}$ had to be selected in case a product is visually standing out among the others, and was determined by visual inspection. Using this technique, the user–item matrix is transformed from a set of $n_{\text{items}} = 4900$ to $n_{\text{in-stock}} = 400$, converting 91% of the transactions conducted with now discontinued products. This effectively increased the user–item matrix occupancy from 0.6% to 5.5%, which is bound to increase over time with inventory rotation.

This also presents an original solution to the cold-start problem for items. Given the large collection of out-of-stock products, a newly added product has a large probability of becoming the in-stock equivalent for at least one of them. When this is the case, the *time translation* algorithm will automatically associate the transaction history of these out-of-stock products to the new one, which allows the latter to be recommended without any business rules.

2.2 *Next-nearest neighbours* & popularity

However, after applying the *time translation*, the items exhibit a traditional long-tail popularity distribution, as seen in Fig. 1(a). Indeed, item popularity is the strongest signal in the data and, while letting the model learn from it allows to predict purchases more accurately, it results in obvious recommendations with poor inventory coverage and diversity [2]. To mitigate the latter issues, the model needs to learn the item–item correlations with the popularity bias removed, so that the recommendations are both original and achieve better inventory coverage by leveraging the non-trivial correlations present in the data.

A first step to circumvent the popularity bias is to consider not only the closest neighbours in the *time translation* scheme, but also the next-nearest neighbours in the feature space. More specifically, instead of considering only the nearest in-stock neighbour, we considered up to five nearest neighbours within a distance $d_0 + \sigma_d$ of the out-of-stock item being mapped, where d_0 is the distance

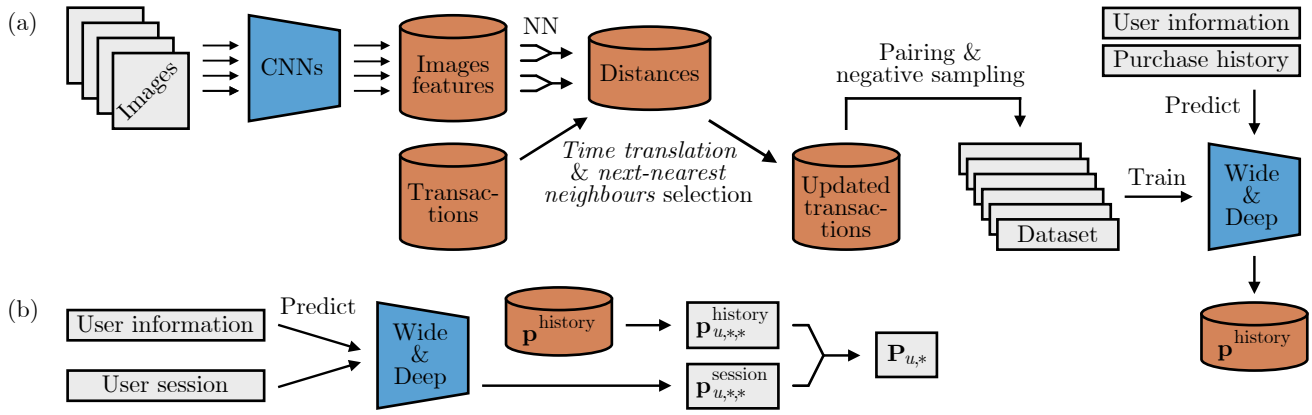

Figure 3: Figure (a) shows the offline process. First, images of all products are encoded in feature vectors using an ensemble of pre-trained CNNs. These vectors are used to compute distances between the products using an approximate nearest neighbours algorithm, and the results are stored in a database. The transactions history of every user is then updated using *time translation* and these distances. This updated transaction history is used to create the dataset, train the model, and produce user-based predictions. The trained model and the predictions are then stored. Figure (b) shows the online process, where the model is queried with the user session activities to create a probability vector, which is added to the one obtained from the purchase history to produce the final scores $\mathbf{P}_{u,*}$ for this user u.

to the nearest neighbour and σ_d is a hyperparameter. The *time translation* scheme with *next-nearest neighbours* selection will then pick the one with the lowest popularity. The effect on the popularity, shown in Fig. 1(a), is reflected by a higher long tail.

2.3 Item–item correlations & probabilities

The training dataset entries contain pairs of items bought together to create the concept of complementary products, similarly to a co-occurrence matrix setting [3]. The dataset is built by enumerating, for every user, all time-ordered combinations of products present in their transaction history.

This choice is motivated by the fact that small retailers tend to have clients with short buying histories that would yield poor performance with collaborative filtering techniques. For instance, in the case of our dataset, most users have less than five items in their history. In this context, it is more interesting to model implicit item–item interactions aggregated across all users. Not only is the latter matrix smaller in the case of our dataset — $n_{\text{items}} = 4900$ vs $n_{\text{users}} = 5700$ — but the *time translation* scheme further reduces its size by a factor of $(n_{\text{items}}/n_{\text{in-stock}})^2$ while it would only reduce the size of a user–item matrix by a factor of $n_{\text{items}}/n_{\text{in-stock}}$. Thus, the item–item formulation makes the sparsity more manageable.

Besides product characteristics describing the pairs of items, the dataset entries contain the time between the two purchases $\Delta t = t_{\text{suggested}} - t_{\text{bought}}$, where $\Delta t \geq 0$. The pairs with Δt greater than a given threshold are discarded, as correlations between items decrease with increasing Δt. This rule is based on the assumption that customer taste and trends evolve substantially within a time window smaller than the threshold.

While the data provides positive labels, learning from implicit item–item interactions requires negative labels, i.e. examples of products not bought together. Drawing items at random from a uniform distribution to generate negatively labelled pairs would under-represent popular items and over-represent unpopular ones, aggravating the popularity bias discussed earlier. To compensate for this bias, we sampled individual items from the popularity distribution to produce an amount of negatively labelled item–item pairs equal to the amount of positively labelled ones [10].

We used the recently published Wide & Deep model as it applies very well to our dataset [1, 6]. The net contribution of the *wide* part of the model is to memorize the frequency at which two products were bought together in the training data. That is exactly what a co-occurrence probability matrix does, which is a natural baseline in an item–item framework. Therefore, since it is jointly trained with the *wide* part, the *deep* part simply improve on this baseline model. Its flexibility add to the resulting model the capability to generalize item–item pair probabilities beyond the co-occurrence probability matrix entries, on top of the capacity to handle extra features. The architecture is shown in Fig. 2.

After training, the model is used offline to generate predictions for all transaction entries in a user history combined with all $n_{\text{in-stock}}$ in–stock items, and this is done for all n_{users} users in the database. The resulting pair probabilities are stored in a tensor $\mathbf{p}$ of size $n_{\text{users}} \times n_{\text{in-stock}} \times n_{\text{items}}$, missing values being implicit zeros. We define the final score $\mathbf{P}_{u,i}$ for a user–item pair as the sum of $\mathbf{p}$ over the user's purchase history[1]:

$$\mathbf{P}_{u,i} = \sum_{j=0}^{n_{\text{items}}} \mathbf{p}_{u,i,j}, \quad \mathbf{P} \in \mathbb{R}^{n_{\text{users}} \times n_{\text{in-stock}}}, \tag{1}$$

[1] We could instead combine the individual predicted probabilities $\mathbf{p}_{u,i,j}$ into a net probability of a given user u buying an item i using

$$1 - \mathbf{P}_{u,i} = \prod_{j=0}^{n_{\text{items}}} \left(1 - \mathbf{p}_{u,i,j}\right).$$

However, since the final score $\mathbf{P}$ is only used to sort the recommendations, we use Eq. 1, which is simpler and provides the same rankings.

where i runs over in–stock products and j runs over all products.

This item–item framework has the added benefit of allowing a real-time update of the recommendations. As a user u navigates the website and shows interest in a product j by visiting its page or adding it into the cart, a column of new probability predictions can be appended to $\mathbf{p}_{u,*,*}$ and added to $\mathbf{P}_{u,*}$. Sorting the updated vector $\mathbf{P}_{u,*}$ allows to obtain new top-k recommendations in real-time without the need to re-train the model.

Additionally, the knowledge of the similarity between all products through the distance metric presents a good opportunity to adjust the scores $\mathbf{P}$ with user feedbacks or business rules. This supposes that we have a rule to establish the correction $\mathbf{a}_{u,j}$ that should be added to the score $\mathbf{P}_{u,j}$ when user u does, for instance, rate an item j. In this case, we can also adjust the scores of products $\{i\}$ in the same category $\vec{c}_j$ as item j according to their distance $\mathbf{d}_{i,j}$:

$$\Delta\mathbf{p}_{u,i,j} = \mathbf{a}_{u,j}\delta_{\vec{c}_i\vec{c}_j}f(\mathbf{d}_{i,j}),\ \mathbf{a} \in \mathbb{R}^{n_{\text{users}}\times n_{\text{items}}},\ \Delta\mathbf{p} \in \mathbb{R}^{n_{\text{users}}\times n_{\text{in-stock}}\times n_{\text{items}}}, \tag{2}$$

where δ_{ij} is the Kronecker delta and $f(\mathbf{d}_{i,j})$ is a function that monotonically decreases from 1 to 0 with increasing $\mathbf{d}_{i,j}$. The score $\mathbf{P}_{u,i}$ will then be the sum of all probabilities from the purchase history $\mathbf{p}_{u,i,*}$ combined with these adjustments $\Delta\mathbf{p}_{u,i,*}$:

$$\mathbf{P}_{u,i} = \sum_{j=0}^{n_{\text{items}}} \mathbf{p}_{u,i,j} + \Delta\mathbf{p}_{u,i,j}. \tag{3}$$

The complete workflow is presented in Fig. 3.

3 EXPERIMENTS AND RESULTS

In order to quantify the advantages of our techniques, we have compared the performance obtained with and without *time translation* and *next-nearest neighbours* selection for the current Wide & Deep model and for a simple co-occurrence probability matrix baseline. Two metrics are provided: the area under the curve (AUC), which assesses the accuracy of the classifier without the need to set a threshold [5], and the top-8 recommendation coverage. Per business rule, an item is defined as covered if it is located within the first eight recommendations for at least one user.

To build the co-occurrence probability matrix model, we first compute the count $\mathbf{C}_{i,j}$ of entries in the training dataset where a pair of items (i, j) were bought together (positively labelled) and the count $\mathbf{D}_{i,j}$ of entries where the same pair of items were not bought together (negatively labelled). We then construct the co-occurrence probability matrix as:

$$\tilde{\mathbf{p}}_{i,j} = \mathbf{C}_{i,j}/\left(\mathbf{C}_{i,j} + \mathbf{D}_{i,j}\right), \tag{4}$$

setting to zero entries with no count. Finally, we return $\tilde{\mathbf{p}}_{i,j}$ as the predicted probability that a suggested item j is bought provided that the user already bought item i.

The results are presented in Table 1. When using the Wide & Deep model, *time translation* increases the test set AUC by 0.032, which is further raised by 0.009 when enabling the *next-nearest neighbours* selection. The former increase is due to the decreased sparsity of the interaction matrix, and the latter one is likely due to the more uniform item distribution obtained, which reduces the amount of item–item co-occurrences that are summed out and lost.

Table 1: Area under the curve (AUC) and coverage obtained with and without *time translation* and/or *next-nearest neighbours* selection. Results are obtained for both Wide & Deep and co-occurrence probability matrix models.

Model	*Time translation*	*Next-nearest neighbours*	AUC	Coverage (%)
Wide & Deep	Yes	Yes	0.761(2)	62.2%
	Yes	No	0.752(2)	45.5%
	No	No	0.713(2)	73.8%
Co-occurrence matrix	Yes	Yes	0.664	—
	Yes	No	0.661	—
	No	No	0.589	—

Also, when using the co-occurrence model — the most popular recommendation algorithm in an item–item setting — the techniques we developed have an even more substantial impact and increase the AUC by 0.075.

The coverage is also significantly increased with the *next-nearest neighbours* selection, growing from 45.5% to 62.2%. This is reflected by a popularity distribution tail remarkably higher, as shown in Fig. 1(b). We also note that the higher coverage obtained without *time translation* comes at the cost of a lower AUC. In this case, it is also likely that the higher sparsity of the data makes the model more prone to produce random predictions, which would increase the coverage with irrelevant recommendations. In contrast, the increased coverage obtained with *next-nearest neighbours* selection comes at no accuracy expense and contains higher quality recommendations.

4 CONCLUSION

The use of features contained in product images to improve the accuracy of recommender systems has been investigated in the past [12, 15, 17], but without addressing the problematic of data sparsity in constantly renewed inventories. Here, we developed such an approach that adequately addresses the latter issue: a *time translation* scheme that maps items out of stock to items in stock based on image similarities. The reduced data sparsity obtained with this scheme resulted in a 0.032 AUC increase when using a Wide & Deep classifier. Furthermore, by systematically selecting the *next-nearest neighbours* in the feature space that are less popular, we also increased the coverage by a substantial 16.7% at no accuracy cost. Our resulting recommender system outperforms the traditional baseline AUC by 0.172, which will result in a dramatic improvement of the recommendations quality and user experience.

While our work specifically treats images, the *time translation* technique is not specific to this type of input and can be transposed to any deep features available. Also, given the dynamic nature of our solutions, it will be of interest to closely follow the evolution of the AUC and coverage over time, as the transaction dataset grows. Finally, since the quantity of out-of-stock items is getting larger, new items are increasingly likely to receive some of the *time translation* mappings, which solves the item cold-start problem.

REFERENCES

[1] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. (2015). https://www.tensorflow.org/

[2] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling Popularity Bias in Learning-to-Rank Recommendation. In *Proceedings of the 11th ACM Conference on Recommender Systems*. ACM, New York, New York, USA, 42–46.

[3] Charu C Aggarwal. 2016. *Recommender Systems: The Textbook* (1st ed.). Springer International Publishing.

[4] Jon Louis Bentley. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* 18, 9 (1975), 509–517.

[5] Andrew P Bradley. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition* 30, 7 (1997), 1145–1159.

[6] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, New York, New York, USA, 7–10.

[7] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, New York, New York, USA, 191–198.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 248–255.

[9] Eui-Hong Sam Han and George Karypis. 2005. Feature-based recommendation system. In *Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM, New York, New York, USA, 446–452.

[10] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, New York, USA, 549–558.

[11] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional Architecture for Fast Feature Embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*. ACM Press, New York, New York, USA, 675–678.

[12] Yushi Jing, David Liu, Dmitry Kislyuk, Andrew Zhai, Jiajing Xu, Jeff Donahue, and Sarah Tavel. 2015. Visual Search at Pinterest. In *the 21th ACM SIGKDD International Conference*. ACM Press, New York, New York, USA, 1889–1898.

[13] Marius Kaminskas, Derek Bridge, Franclin Foping, and Donogh Roche. 2015. Product Recommendation for Small-Scale Retailers. In *16th International Conference on Electronic Commerce and Web Technologies*. 17–29.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*. 1097–1105.

[15] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, New York, USA, 43–52.

[16] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[17] Yong-Siang Shih, Kai-Yueh Chang, Hsuan-Tien Lin, and Min Sun. 2018. Compatibility Family Learning for Item Recommendation and Generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*. 2403–2410.

[18] Brent Smith and Greg Linden. 2017. Two Decades of Recommender Systems at Amazon.com. *IEEE Internet Computing* 21, 3 (2017), 12–18.

[19] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going Deeper with Convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition*. 1–9.

[20] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale Commodity Embedding for E-commerce Recommendation in Alibaba. (March 2018). arXiv:1803.02349

[21] Shuai Zhang, Lina Yao, and Aixin Sun. 2017. Deep Learning based Recommender System: A Survey and New Perspectives. (July 2017). arXiv:1707.07435