# History-Augmented Collaborative Filtering for Financial Recommendations

Baptiste Barreau
Université Paris-Saclay, CentraleSupélec
Mathématiques et Informatique pour la Complexité et les Systèmes, Chair of Quantitative Finance
Gif-sur-Yvette, France
BNP Paribas Corporate and Institutional Banking
Global Markets Data & Artificial Intelligence Lab
Paris, France
baptiste.barreau@bnpparibas.com

Laurent Carlier
BNP Paribas Corporate and Institutional Banking
Global Markets Data & Artificial Intelligence Lab
Paris, France
laurent.carlier@bnpparibas.com

## ABSTRACT

In many businesses, and particularly in finance, the behavior of a client might drastically change over time. It is consequently crucial for recommender systems used in such environments to be able to adapt to these changes. In this study, we propose a novel collaborative filtering algorithm that captures the temporal context of a user-item interaction through the users' and items' recent interaction histories to provide dynamic recommendations. The algorithm, designed with issues specific to the financial world in mind, uses a custom neural network architecture that tackles the non-stationarity of users' and items' behaviors. The performance and properties of the algorithm are monitored in a series of experiments on a G10 bond request for quotation proprietary database from BNP Paribas Corporate and Institutional Banking.

## CCS CONCEPTS

• **Computing methodologies** → *Machine learning*; • **Information systems** → Information retrieval; **Recommender systems**; **Business intelligence**.

## KEYWORDS

matrix factorization, collaborative filtering, context-aware, time, neural networks

**ACM Reference Format:**
Baptiste Barreau and Laurent Carlier. 2020. History-Augmented Collaborative Filtering for Financial Recommendations. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20), September 22–26, 2020, Virtual Event, Brazil.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3383313.3412206

## 1 INTRODUCTION

In a financial market, liquidity is provided by *market makers*, whose role is to constantly offer both buy (*bid*) and sell (*ask*) prices. Market makers benefit from the difference between the two, called the *bid-ask spread*. Corporate and institutional banks such as BNP Paribas CIB play the role of market makers in financial markets across many asset classes and their derivatives. When a client wants to trade a financial product, she either requests prices on an electronic platform where many different market makers operate in a process called a *request for quotation* (RFQ), or contact a salesperson of a bank. Reciprocally, salespeople can also directly contact clients and suggest relevant trade ideas to them, e.g., financial products held by the bank and on which it might offer a better price than its competitors. Proactive salespeople, which are particularly important for the bank, help manage financial inventories to minimize the risk to which the bank is exposed, and serve better their clients when correctly anticipating their needs. Providing salespeople with an RFQ recommender system would support their proactivity by allowing them to navigate the complexity of the markets more easily. Our goal is to design a financial recommender system that suits the particularities of the financial world to assist salespeople in their daily tasks. RFQ recommendation is an implicit feedback problem, as we do not explicitly observe clients' opinions about the products they request. Implicit feedback is a classic recommender system setup already addressed by the research community, e.g., in [9]. The financial environment, however, brings about specific issues that require attention. To that end, the algorithm we introduce here has three main aims:

- **To incorporate time.** In a classic e-commerce environment, leaving aside popularity and seasonal effects, recommendations provided at a given date may remain relevant for a couple of months, since users' shopping tastes do not markedly evolve with time. In the financial world, a user is interested in a financial product at a given date not only because of the product's intrinsic characteristics but also because of its current market conditions. Time is consequently crucial for RFQ prediction and should be taken into account in a manner that allows for future predictions.
- **To obtain dynamic embeddings.** Being able to capture how clients (resp. financial products) relate to each other and how this relationship evolves with time is of great interest for business, as it allows getting a deeper understanding of the market. One of our goals is consequently to keep the global architecture of matrix factorization algorithms, where


*RecSys '20, September 22–26, 2020, Virtual Event, Brazil*

ACM ISBN 978-1-4503-7583-2/20/09…$15.00
https://doi.org/10.1145/3383313.3412206

the recommendation score of a *(client, product)* couple is given by the scalar product of their latent representations.

- **To symmetrize users and items.** Classic recommender systems focus on the user-side. However, the product-side is also of interest in the context of a corporate bank. To control her risk, a market maker needs to control her positions, i.e., make sure she does not hold a given position for too long. Consequently, coming up with a relevant client for a given asset is equally important, and the *symmetry of clients and assets* will be integrated here in both our algorithms and evaluation strategies.

We tackle these goals with a context-aware recommender system that only uses client-product interactions as its signal source. In this work, the context is dynamic, and is inferred at a given time from the previous interactions of clients and products. The terms clients and users (resp. financial products/assets and items) will be used interchangeably to match the vocabulary used in recommender systems literature.

## 2 RELATED WORK

The work presented in this article is a context-aware and time-dependent recommender system, which are both active areas of research [1, 17]. Notably, in [7], the recommendations of a bandit algorithm are dynamically adapted to contextual changes. In [8], the recommendations of a hierarchical hidden Markov model are contextualized based on the users' feedback sequences. Down-weighing past samples in memory-based collaborative filtering algorithms helps better match temporal dynamics [5]. The latent factors of a matrix factorization can also directly include temporal dynamics [10]. Temporal probabilistic matrix factorization [25] introduces dynamics of users' latent factors with a time-invariant transition matrix that can be computed using Bayesian approaches, thereby extending probabilistic matrix factorization [14]. It is also possible to enhance latent factor models with Kalman filters [16] or recurrent neural networks [21] to capture the dynamics of rating vectors. Tensor factorization models handle time by considering the user-item co-occurrence matrix as a three-dimensional tensor where the additional dimension corresponds to time [22, 24], and where temporal dynamics can be accounted for using recurrent neural networks [22] — these approaches, however, do not allow for future predictions. Using historical data to introduce dynamics in a neural network recommender system was done in [3], where users are assimilated to their items' histories. The Caser algorithm [18] uses convolutional filters to embed histories of previous items and provide dynamic recommendations to users. The algorithm introduced in this work is to some extent reminiscent of graph neural networks [6] and their adaptation to recommendation [19]. Using time to enhance recommendations with random walks on bipartite graphs was explored in [23]. How graph neural networks behave with dynamic bipartite graphs is to the best of our knowledge yet to be discovered and could lead to an extension of this work. A first attempt at financial products recommendation was made in [20] with the particular example of corporate bonds.

## 3 METHODOLOGY

We introduce a neural network architecture that aims at producing recommendations through dynamic embeddings of users and items, that we call *History-augmented Collaborative Filtering* (HCF). Let $U$ be the set of all users and $I$ the set of all items. Let us note $x_u \in \mathbb{R}^d$ for $u \in U$, $x_i \in \mathbb{R}^d$ for $i \in I$ the static $d$-dimensional embeddings of all the users and items we consider. Let $t \in [\![0\,;\infty]\!]$ be a discrete time-step, taken as days in this work. For a given $u$, at a given $t$, we define $\mathbf{h_u^t}$ as the items' history of $u$, i.e., the set of items found to be of interest to $u$ in the past — we respectively define $\mathbf{h_i^t}$ as item $i$ users' history. We use here as histories the last $n$ events that happened strictly before $t$ to either user $u$ or item $i$. If at a given $t$ we observe $0 \leq n' < n$ previous events, histories are only formed of those $n'$ events. Users are consequently on a same event scale — high-activity users' histories may span a couple of days, whereas low-activity ones may span a couple of months (resp. for items). Histories formed of items/users of interest in a past window of fixed size were also tried, but led to inferior performance.

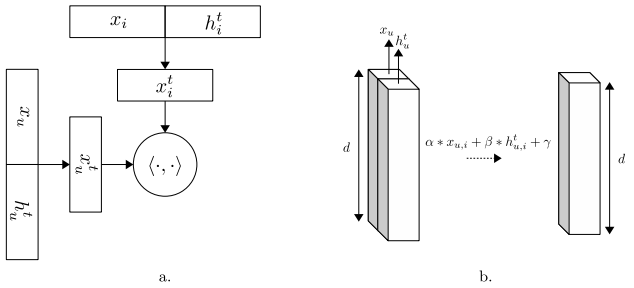


**Figure 1: a. Global architecture of the HCF network, composed of two symmetric user and item blocks. b. Illustration of the application of a one-dimensional convolution filter of kernel size 1 along the embedding dimension axis of our inputs $x_u$ and $h_u^t$ for a user $u \in U$, where $\alpha, \beta, \gamma$ are the parameters of the convolution filter and $l \in [\![1; d]\!]$. The same computation holds for items $i \in I$ respectively.**

Figure 1.a shows the global architecture of the HCF network. It is composed of two symmetric blocks — a *user block* and an *item block*. At a given time $t$, the user block of HCF produces dynamic user embeddings $x_u^t$ using as inputs static embeddings of users $x_u$ and their corresponding histories' embeddings $h_u^t$, defined as $h_u^t = \frac{1}{|\mathbf{h_u^t}|} \sum_{i \in \mathbf{h_u^t}} x_i$. If $\mathbf{h_u^t}$ is empty, we use $h_u^t = 0$. Respectively, the item block produces dynamic item embeddings $x_i^t$, and the score of a *(u,i)* couple at a given $t$ is given by the scalar product of their dynamic embeddings. In each block, dynamic embeddings are computed using a network of one-dimensional convolution filters of kernel size 1 along the embedding dimension axis, considering user and history embeddings as channels (see Fig. 1.b). Convolutions were chosen because we empirically found that all architectures performing computations involving both $x_{u,l}$ and $h_{u,l'}^t$ with $l \neq l' \in [\![1; d]\!]$ the $l$-th component of the embedding systematically led to poorer performance than a linear component-wise combination of $x_u$ and $h_u^t$. The chosen convolutions can be seen as a variation

of the linear component-wise combination with shared parameters across all components, and allow for network depth.

The network is trained using the *bayesian personalized ranking* (BPR) loss [15], a surrogate of the ROC AUC score [12]. It is defined in [15] as $L_{BPR} = -\sum_{(u,i,j)\in D} \ln \sigma(x_{uij})$, where $D = \{(u,i,j)|i \in I_u^+ \wedge j \in I \backslash I_u^+\}$ with $I_u^+$ the subset of items that were of interest for user $u$ in the considered dataset, and $x_{uij} = x_{ui} - x_{uj}$, with $x_{ui}$ the score of the $(u,i)$ couple. $\sigma$ is the sigmoid function, defined as $\sigma(x) = 1/(1+e^{-x})$. The underlying idea is to rank items of interest for a given $u$ higher than items of no interest for that $u$, and $D$ corresponds to the set of all such possible pairs for all users appearing in the considered dataset. As $D$ grows exponentially with the number of users and items considered, it is usual to approximate $L_{BPR}$ with negative sampling [13].

In our proposed methodology, scores become time-dependent. Data samples are therefore not seen as couples, but as triplets $(t,u,i)$. To enforce user-item symmetry in the sampling strategy, we define

$$D_u^t = \{(t,u,i,j)|i \in I_u^{t,+} \wedge j \in I \backslash \mathbf{h_u^t}\}$$
$$D_i^t = \{(t,u,v,i)|u \in U_i^{t,+} \wedge v \in U \backslash \mathbf{h_i^t}\}$$

with $I_u^{t,+}$ the subset of items that had a positive interaction with user $u$ at time $t$, and $U_i^{t,+}$ the subset of users that had a positive interaction with item $i$ at time $t$. For a given positive triplet $(t,u,i)$, we sample either a corresponding negative one in $D_u^t$ or $D_i^t$ with equal probability. Note that considering samples as triplets adds a sampling direction, as a couple that was active at a time $t$ may no longer be active at other times $t+t'$ or $t-t$", $t', t" > 0$. Such sampling strategies will be more extensively studied in further iterations of this work.

## 4 EXPERIMENTS

We conduct a series of experiments to understand the behavior of the proposed HCF algorithm on a proprietary database of RFQs on governmental bonds from the G10 countries. This database, which describes every day the RFQs performed by the clients of the bank on G10 bonds, accounts for hundreds of clients and thousands of bonds and ranges from 08/01/2018 to 09/30/2019. On average, we observe tens of thousands of user-item interactions every month. We examine here the performance of our proposal in comparison to benchmark algorithms in two experiments. Benchmark algorithms, chosen for their respect of the aims outlined in Section 1, are a historical baseline and two matrix factorization algorithms trained using a confidence-weighted euclidean distance [9] and the BPR objective [15], that we respectively call *MF - implicit* and *MF - BPR*. MF - implicit is trained with gradient descent, and we adopt for MF - BPR the symmetrical sampling strategy outlined in Section 3. The historical baseline scores a *(user, item)* couple with their observed number of interactions during the considered training period. In this section, the performance of our models is evaluated using mean average precision (mAP) [12], defined as $\text{mAP} = 1/|Q| * \sum_{q\in Q} \text{AP}(q)$ where Q is the set of *queries* to the recommender system, and $\text{AP}(q)$ is the average precision score of a given query $q$. To monitor the performance of our algorithms on both the user- and item-sides, we define two sets of queries over which averaging:

- ***User-side queries.*** Queries correspond to the recommendation list formulated every day for all users;
- ***Item-side queries.*** Queries correspond to the recommendation list formulated every day for all items.

These two query sets lead to user- and item-side mAPs that we summarize with a harmonic mean in a symmetrized mAP score used to monitor all the following experiments, as $\text{mAP}_{sym} = \frac{2*\text{mAP}_u*\text{mAP}_i}{\text{mAP}_u+\text{mAP}_i}$. The user- and item-sides equally contribute to the symmetrized score since both sides equally matter in our financial context, as seen in Section 1. The mAP scoring perimeter corresponds to the Cartesian product of all the users and items observed in the training period[1].

### 4.1 Evolution of forward performance with training window size

This experiment aims at showing how benchmark algorithms and our HCF proposal behave with regards to stationarity issues. We already advocated the importance of time in the financial setup — taking the user side, clients' behavior is *non-stationary*, owing to the non-stationarity of the financial markets themselves but also externalities such as punctual needs for liquidity, regulatory requirements, … In machine learning, this translates into the fact that the utility of past data decreases with time. However, machine learning and more particularly deep learning algorithms work best when provided with large datasets [11]: there is an apparent trade-off between non-stationarity and the need for more training data. Our goal in this experiment is to show that introducing time in the algorithm helps to reduce this trade-off.

To prove this, we examine the evolution of forward performance as the training window size grows. We split the G10 bonds RFQ dataset into three contiguous parts — a train part that ranges from up to 08/01/2018 to 07/31/2019 (up to one year), a validation part from 08/01/2019 to 08/30/2019 (one month) and a test part from 09/01/2019 to 09/30/2019 (one month). Validation is kept temporally separated from the training period to avoid signal leakages [4], and is taken forward to match business needs. For all the considered algorithms, we train an instance of each on many training window sizes ranging from a week to a year using carefully hand-tuned hyperparameters and early stopping, monitoring validation symmetrized mAP.

We see in Fig. 2 that all benchmark algorithms present a bell-shaped curve. They attain a peak after which their performance only degrades as we feed these models more data, corroborating the non-stationarity vs. amount of data trade-off. On the contrary, HCF only gets better with training data size. Notably, HCF 12m which obtained best validation performance used $n = 20$ and blocks with two hidden layers and ReLU activations. To show that these bell-shaped curves are not an artifact of the hyperparameters chosen in the previous experiment, we conduct a systematic hyperparameter search for multiple training window sizes using a combination of a hundred trials of random search [2] and hand-tuning. The optimal sets of hyperparameters for each considered window size are then used as in the previous experiment to obtain graphs of their

[1] This scoring perimeter proved to be the fairest with regard to all considered models, as a model is consequently scored only on what it can score and nothing else. Fixing the perimeter for all algorithms to the Cartesian product of all the users and items observed in the maximal training window and attributing the lowest possible score to otherwise unscorable couples only lowers the mAP scores of candidate algorithms that cannot obtain satisfactory results on such window sizes.

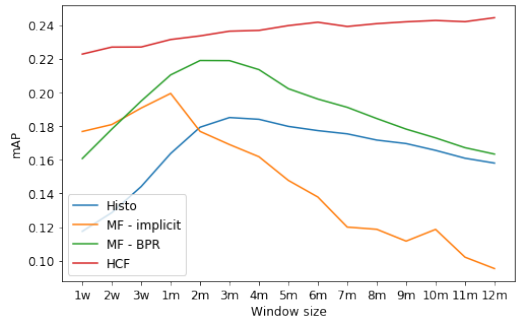


**Figure 2: Evolution of validation symmetrized mAP with training window size. Whereas benchmark algorithms seem to have an optimal training window size, our HCF proposal keeps improving with the training window size.**

validation mAP scores against the training window size. Results are shown in Figure 3. MF - BPR 6m and 12m optimal hyperparameters happened to coincide, and their results are consequently shown on the same graph.

We see here that for both MF algorithms, hyperparameters optimized for many different window sizes seem to agree on optimal window size, respectively around one month and two months, with slight variations around these peaks. Consequently, bell shapes are inherent to these algorithms, which proves their non-stationarity vs. data size trade-off. To obtain test performances that are not penalized by the discontinuity of the training and test windows, we retrain all these algorithms with the best hyperparameters and window size found for the validation period on dates directly preceding the test period, using the same number of training epochs as before. The performances of all the considered algorithms are reported in Table 1.

**Table 1: Window size study — symmetrized mAP scores, in percentage.**

| Algorithm | Window | Valid mAP | Test mAP |
|---|---|---|---|
| **Historical** | 3m | 18.51 | 16.86 |
| **MF - implicit** | 1m | 19.94 | 19.24 |
| **MF - BPR** | 2m | 21.89 | 20.05 |
| **HCF** | 12m | **24.43** | **25.02** |

It appears that our HCF algorithm, augmented with the temporal context, obtains better performances on both validation and test periods than the static benchmark algorithms. Consequently, introducing temporal dynamics is essential in the financial setup that we consider.

## 4.2 Evolution of forward performance with time

It follows from the previous experiment that our benchmark algorithms cannot make proper use of large amounts of past data and have to use short training window sizes to obtain good performances compared to historical models. Moreover, the results of these algorithms are less stable in time than HCF results. Fig. 4 visualizes the results reported on Table 1 on a daily basis for all the considered algorithms.

We see a downward performance trend for all the benchmark algorithms — the further away from the training period, the lower the daily symmetrized mAP. On the contrary, HCF has stable results over the whole test period: introducing temporal context through user and item histories hinders the non-stationarity effects on forward performances. The results from Section 4.1 and the observed downward performance trend consequently suggest that benchmark models need frequent retraining to remain relevant regarding future interests. A simple way to improve their results is to retrain these models on a daily basis with a constant, sliding window size $w$ — predictions for each day of the testing period are made using a model trained on the $w$ previous days. Each model uses here the number of epochs and hyperparameters determined as best on the validation period in the previous experiment. Results of these *sliding* models are shown in Table 2, where HCF results corresponds to the previous ones.

**Table 2: Sliding study — symmetrized mAP scores, expressed in percentage.**

| Algorithm | Window | Test mAP |
|---|---|---|
| **Historical (sliding)** | 3m | 20.32 |
| **MF - implicit (sliding)** | 1m | 24.27 |
| **MF - BPR (sliding)** | 2m | 24.46 |
| **HCF** | 12m | **25.02** |

We see that both MF - implicit and MF - BPR significantly improved their results compared to their static versions from Table 1, but are still below the results of HCF trained on 12 months. Consequently, our HCF proposal is inherently more stable than our benchmark algorithms and captures time in a more efficient manner than their daily retrained versions.

## 5 CONCLUSION

We introduce a novel HCF algorithm, a time-aware recommender system that uses user and item histories to capture the dynamics of the user-item interactions and that provides dynamic recommendations. In the context of financial G10 bonds RFQ recommendations, we show that for classic matrix factorization algorithms, a trade-off exists between the non-stationarity of users' and items' behaviors and the size of the training datasets. This trade-off is overcome with history-augmented embeddings. Moreover, these embeddings outperform sliding versions of classic matrix factorization algorithms and prove to be more stable predictors of the future interests of the users and items. Further research on the subject will include a more thorough investigation of alternative histories' embeddings formulations and time-aware sampling strategies. Finally, the HCF algorithm could be applied beyond the financial world to tasks where temporal dynamics drive users' behaviors, e.g., music and movie recommendation, where the current mood of a user highly influences her next decisions.

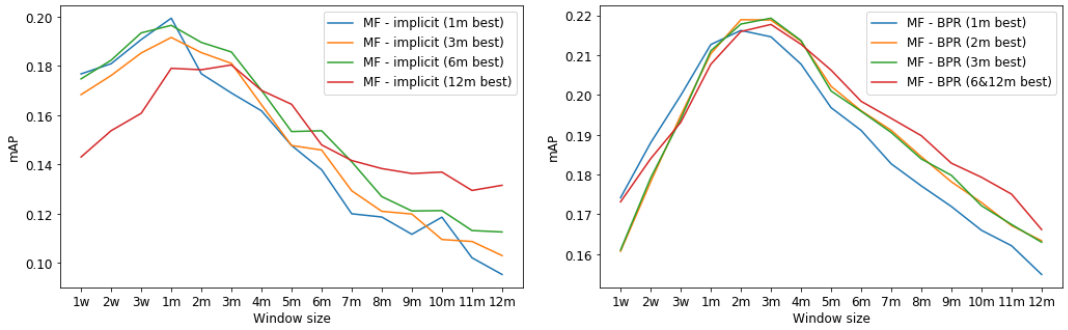


**Figure 3: Evolution of validation symmetrized mAP with training window size. *Left:* Optimal MF - implicit for the 1m, 3m, 6m and 12m windows. A consensus arises around the 1m window. *Right:* Optimal MF - BPR for the 1m, 2m, 3m, 6m and 12m windows. A consensus arises around the 2m-3m windows, slightly favouring the 2m window.**

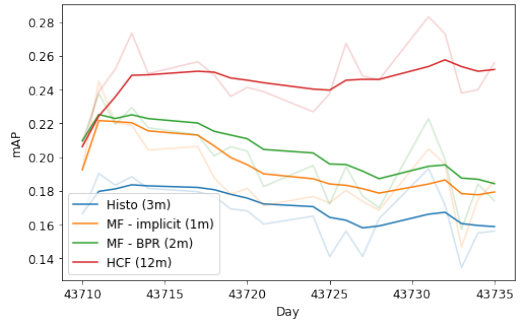


**Figure 4: Daily evolution of symmetrized mAP during the test period. Light shades correspond to the true daily symmetrized mAP values, and dark ones to exponentially weighted averages ($\alpha = 0.2$) of these values.**

## ACKNOWLEDGMENTS

This work has been conducted under the French CIFRE Ph.D. program, in collaboration between the MICS Laboratory at CentraleSupélec and BNP Paribas CIB Global Markets. We thank Dan Sfedj and Camille Garcin for their work on early versions of the HCF algorithm, and Damien Challet, Sarah Lemler and Julien Dinh for helpful discussions and feedback on drafts of this work.

## REFERENCES

[1] Gediminas Adomavicius, Bamshad Mobasher, Francesco Ricci, and Alexander Tuzhilin. 2011. Context-aware recommender systems. *AI Magazine* (2011), 67–80.
[2] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13, Feb (2012), 281–305.
[3] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for YouTube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 191–198.
[4] Marcos Lopez De Prado. 2018. *Advances in financial machine learning*. John Wiley & Sons, Chapter 7.
[5] Yi Ding and Xue Li. 2005. Time weight collaborative filtering. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*. 485–492.
[6] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.
[7] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2014. Context adaptation in interactive recommender systems. In *Proceedings of the 8th ACM Conference on Recommender Systems*. 41–48.
[8] Mehdi Hosseinzadeh Aghdam, Negar Hariri, Bamshad Mobasher, and Robin Burke. 2015. Adapting recommendations to contextual changes using hierarchical hidden Markov models. In *Proceedings of the 9th ACM Conference on Recommender Systems*. 241–244.
[9] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 8th IEEE International Conference on Data Mining*. Ieee, 263–272.
[10] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 447–456.
[11] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.
[12] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. 2010. *Introduction to information retrieval*. Vol. 16. Cambridge University Press, Chapter 8.4, 100–103.
[13] 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.
[14] Andriy Mnih and Russ R. Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*. 1257–1264.
[15] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 452–461.
[16] Purnamrita Sarkar, Sajid M Siddiqi, and Geogrey J Gordon. 2007. A latent space approach to dynamic embedding of co-occurrence data. In *Artificial Intelligence and Statistics*. 420–427.
[17] Yue Shi, Martha Larson, and Alan Hanjalic. 2014. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)* 47, 1 (2014), 1–45.
[18] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 565–573.
[19] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM*

*SIGIR Conference on Research & Development in Information Retrieval*. 165–174.

[20] Dominic Wright, Luca Capriotti, and Jacky Lee. 2018. Machine learning and corporate bond trading. *Algorithmic Finance* 7, 3-4 (2018), 105–110.

[21] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*. 495–503.

[22] Xian Wu, Baoxu Shi, Yuxiao Dong, Chao Huang, and Nitesh Chawla. 2018. Neural Tensor Factorization. *arXiv preprint arXiv:1802.04416* (2018).

[23] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. 2010. Temporal recommendation on graphs via long-and short-term preference fusion. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 723–732.

[24] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime G Carbonell. 2010. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, 211–222.

[25] Chenyi Zhang, Ke Wang, Hongkun Yu, Jianling Sun, and Ee-Peng Lim. 2014. Latent factor transition for dynamic collaborative filtering. In *Proceedings of the 2014 SIAM International Conference on Data Mining*. 452–460.