ИНФОРМАТИКА

Известия Саратовского университета. Новая серия. Серия: Математика. Механика. Информатика. 2022. Т. 22, вып. 1. С. 103–111
Izvestiya of Saratov University. Mathematics. Mechanics. Informatics, 2022, vol. 22, iss. 1, pp. 103–111
https://mmi.sgu.ru
https://doi.org/10.18500/1816-9791-2022-22-1-103-111

Article

Attention based collaborative filtering

A. I. Romanov✉, **I. A. Batraeva**

Saratov State University, 83 Astrakhanskaya St., Saratov 410012, Russia

Aleksey I. Romanov, romanow.lesha2013@yandex.ru, https://orcid.org/0000-0002-0136-1954
Inna I. Batraeva, BatraevaIA@info.sgu.ru, https://orcid.org/0000-0002-6539-8473

Abstract. Attention mechanism invention was an important milestone in the development of the Natural Language Processing domain. It found many applications in different fields, like churn prediction, computer vision, speech recognition, and so on. Many state-of-the-art models are based on attention mechanisms, especially in NLP. As this technique is very powerful, we decided to investigate its application in solving a collaborative filtering problem. In this paper, we propose a standard framework for developing a recommender system engine based on transformer architecture. We could not reproduce current state-of-the-art results on MovieLens datasets, but in our implementation attention based model achieves competitive scores on MovieLens 1M and MovieLens 10M datasets.

Keywords: collaborative filtering, attention mechanism, deep learning, transformer, graph attention network

For citation: Romanov A. I., Batraeva I. A. Attention based collaborative filtering. *Izvestiya of Saratov University. Mathematics. Mechanics. Informatics*, 2022, vol. 22, iss. 1, pp. 103–111. https://doi.org/10.18500/1816-9791-2022-22-1-103-111

Научный отдел

Научная статья
УДК 004.032.26

Коллаборативная фильтрация с механизмом внимания

А. И. Романов✉, **И. А. Батраева**

Саратовский национальный исследовательский государственный университет имени Н. Г. Чернышевского, Россия, 410012, г. Саратов, ул. Астраханская, д. 83

Романов Алексей Иванович, магистр компьютерных наук, romanow.lesha2013@yandex.ru, https://orcid.org/0000-0002-0136-1954
Батраева Инна Александровна, кандидат физико-математических наук, заведующий кафедрой технологий программирования, BatraevaIA@info.sgu.ru, https://orcid.org/0000-0002-6539-8473

Аннотация. Изобретение механизма внимания в нейронных сетях стало важной вехой в развитии области обработки естественного языка. Оно получило множество приложений в различных областях, таких как прогнозирование оттока, компьютерное зрение, распознавание речи и т. д. Многие современные модели основаны на механизме внимания, например архитектура трансформера. Поскольку этот метод уже продемонстрировал свою эффективность, было решено изучить его применение при решении задачи коллаборативной фильтрации. В статье предлагается реализация механизма рекомендательной системы, основанного на архитектуре трансформера. Также в работе приведены результаты сравнительных экспериментов с классическими алгоритмами рекомендательных систем на общедоступном наборе данных.

Ключевые слова: коллаборативная фильтрация, механизм внимания, глубокое обучение, трансформеры, графовые нейронные сети с вниманием

Для цитирования: *Romanov A. I., Batraeva I. A.* Attention based collaborative filtering [*Романов А. И., Батраева И. А.* Коллаборативная фильтрация с механизмом внимания] // Известия Саратовского университета. Новая серия. Серия: Математика. Механика. Информатика. 2022. Т. 22, вып. 1. С. 103–111. https://doi.org/10.18500/1816-9791-2022-22-1-103-111

Introduction

Due to the exponential growth of the web information, web applications face new challenges in providing the best user experience service. Recommender systems are getting to be one of the core components of the web applications, where personalisation may increase users engagement or conversion rate. The most popular use cases are related to recommendations to buy a product, based on the user's historical purchases. Another famous application is a news feed generation on social networks. Recommendations of the entertainment content are also extremely powerful opportunities for recommender systems utilization. For example, in 2016 Netflix reported [1] that its recommender system influenced roughly 80% of streaming hours on the site and further estimated the value of the system at over $1B annually. There are three main types of recommender systems [2]:

- collaborative filtering;
- content-based;
- hybrid methods.

Each of the methods has its own advantages and disadvantages. However, because of its relative simplicity of implementation, currently, the most popular technique is collaborative filtering. Collaborative filtering assumes that the model learns user's preferences based on its previous historical interactions. The common output of collaborative filtering models is users embedding matrix U and items embeddings matrix V. Where embedding means: vector representation of the object. This framework is influenced by a cold start issue: more than half of users and items have very little interaction history and it leads to noisy predictions for them. To address this problem, usually additional user and items features may be utilized as a useful signal, which will improve prediction accuracy for cold objects. This kind of architecture is called a hybrid recommender system. In this paper, we will not focus on content-based and hybrid methods.

1. Problem statement

A collaborative filtering method can be represented as a matrix factorization problem. Given a log of users and items interactions history. Each interaction is represented by triplets: (u_i, i_j, r_{ij}), where r_{ij} is its rating, which was given by user u_i to item i_j. This log can be represented by interaction matrix M. Each row of the matrix is associated with the user, and each column is associated with the item. Each matrix cell will be a rating r_{ij}. In most cases, this matrix has around a 95% of sparsity rate [3], which means that most of the matrix elements will be empty. We have only partial information about the cells of this matrix, based on explicit or observed customer behavior. Explicit behavior may be a product rating given by a customer. Observed behavior tries to deduce how much a customer likes the product by implicit signals, for example, when a customer views a product, adds it to their cart, or purchases it. Our goal is to build a model that can predict the values of the empty cells of this interaction matrix. We try to approximate the interaction matrix as a product of two matrices of lower dimensions, user factors, and item factors: $M = U * V$. The scalar product of a row of matrix U and a column of matrix V gives a predicted item rating for the missing cells. Predictions for known items should be as close to the ground truth as possible. We fit those two matrices with known data using optimization algorithms.

2. Traditional collaborative filtering algorithms

2.1. Latent factor model

Latent factor models map both users and items to a joint low-dimensional latent space, where the user-item preference score is estimated by vector inner product.

We denote user latent vectors as $U = [u_1, ..., u_N] \in \mathbb{R}^{N*K}$ and item latent vectors as $V = [v_1, ..., v_Q] \in \mathbb{R}^{Q*K}$, where $K < \min(N, Q)$ is the latent feature dimension. The preference score $\hat{r}_{ij}$ is estimated as:

$$\hat{r}_{ij} = \langle u_i, v_j \rangle = u_i^T v_j.$$

Xiangnan et al. [4] defined the objective function as regularized squared loss on observed ratings:

$$\arg\min(U, V) \sum_{(i,j)\in Y} (r_{ij} - \hat{r}_{ij}) + \lambda \left(\|U\|^2 + \|V\|^2 \right),$$

where λ controls the strength of regularization, which is usually an L_2 norm to prevent overfitting, Y denotes the set of observed interactions. After we obtain the optimized user and item vectors, the recommendation is then reduced to a ranking problem according to the estimated scores $\hat{r}_{ij}$.

2.2. Weighted regularized matrix factorization

It is problematic to apply traditional matrix factorization to implicit feedback, because we only observe positive feedback ($e.g., r_{ij} = 1,\ \forall r_{ij} \in Y$). We cannot ignore the unobserved user-item interactions, otherwise it will lead to trivial but useless solutions (e.g., collapsing all the latent vectors to a single point). Also, we cannot assume these unobserved interactions as negative either, as we do not know the fact that these interactions did not happen, was because the user did not like the item or the user was not aware of it. To address these issues, Hu et al. and Pan et al. proposed weighted regularized matrix factorization (WRMF) [5] that includes all the unobserved user-item interactions as negative samples and uses a case weight c_{ij} to reduce the impact of these uncertain samples, i.e.:

$$\arg\min(U,V) \sum_{(i,j)\in M} c_{i,j}(r_{ij} - \hat{r}_{ij}) + \lambda \left(\|U\|^2 + \|V\|^2\right),$$

where M denotes a set of observed and unobserved interactions, case weight c_{ij} is larger for observed positive feedback and smaller for unobserved interactions.

2.3. Bayesian Personalized Ranking

BPR is a well-known framework for addressing the implicitness in CF. Instead of point-wise learning as in WRMF, BPR models a triplet of one user and two items, where one of the items is observed and the other one is not. Specifically, from the user-item matrix R, if an item j has been viewed by user i, then it is assumed that the user prefers this item over all the other unobserved items.

The optimization objective for BPR is based on the maximum posterior estimator [6]. In particular, by applying the above latent factor models, a widely used BPR model is given as:

$$\arg\min(U,V) \sum_{(i,j)\in Y} -\ln \delta(\hat{r}_{ij} - \hat{r}_{ik}) + \lambda \left(\|U\|^2 + \|V\|^2\right),$$

where δ is the logistic sigmoid function and λ is a regularization parameter. The training data Y_B is generated as:

$$Y_B = \{(i,j,k)|j \in Y(i) \wedge k \in I \backslash Y(i)\},$$

where I denotes the set of all items in the dataset and $Y(i)$ represents the set of items that are interacted by the i-th user. The semantics of $(i,j,k) \in Y_B$ is that item j is assumed to be preferable over k by user i.

2.4. Neural collaborative filtering

One of the first attempts to utilize neural networks to address a collaborative filtering task was the neural collaborative filtering approach. By replacing the inner product with a neural architecture the model learns how to approximate user-item interaction function.

Model structure. The Embedding layer follows the input layer; it is a fully connected layer that projects the sparse representation to a dense vector. The obtained user (item) embedding can be seen as the latent vector for a user (item) in the context of the latent factor model. The user embedding and item embedding are then fed into a multi-layer neural architecture, which the authors term as neural collaborative filtering layers, to map the latent vectors to prediction scores. The final output layer is the predicted score $\hat{r}_{ij}$, and training is performed by minimizing the pointwise loss between $\hat{r}_{ij}$ and its target value r_{ij}.

NCF's predictive model:

$$\hat{r}_{ij} = f(P^T v_i^U, Q^T v_j^I | P, Q, \theta_f),$$

where $P \in \mathbb{R}^{M*K}$ and $Q \in \mathbb{R}^{N*K}$ denote the latent factor matrix for users and items, respectively; and θ_f denotes the model parameters of the interaction function f. Since the function f is defined as a multi-layer neural network, according to [7] it can be formulated as:

$$f(P^T v_i^U, Q^T v_j^I) = \phi_{out}\left(\phi_x\left(\cdots\left(\phi_2\left(\phi_1(P^T v_i^U, Q^T v_j^I)\right)\right)\right)\right),$$

where ϕ_{out} and ϕ_x respectively denote the mapping function for the output layer and x-th neural collaborative filtering (CF) layer and there are X neural CF layers in total.

To learn model parameters, existing pointwise methods largely perform a regression with squared loss:

$$L = \sum_{(i,j) \in Y \cup Y^-} w_{ij}(r_{ij} - \hat{r}_{ij})^2,$$

where Y denotes the set of observed interactions, Y^- denotes the set of negative instances, which can be all unobserved interactions and w_{ij} is a hyperparameter denoting the weight of training instances (u, i).

3. Related work

3.1. Attention mechanism

Attention mechanism has become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences. Innovation of the transformer model was based on the assumption that it is not necessary to use recurrent neural networks in conjunction with attention to achieve state-of-the-art performance. New model architecture eschews recurrence and instead relies entirely on an attention mechanism to draw global dependencies between input and output.

The Transformer allows for significantly more parallelization and can reach a new state of the art performance on a wide range of NLP tasks using stacked self-attention and pointwise, fully connected layers for both the encoder and decoder.

Encoder: The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position wise fully connected feed-forward network.

Decoder: The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack.

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. The input consists of queries and keys of dimension d_k and values of dimension d_v.

According to [8], attention mechanism can be computed as a dot product of the query with all keys, divide each by $\sqrt{d_k}$, and apply a $Softmax$ function to obtain the weights on the values:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

where Q is a matrix of queries, K is a matrix of keys, and V is a matrix of values. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O,$$

where

$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right).$$

The projections are parameter matrices:

$$W_i^Q \in \mathbb{R}^{d_{model} * d_q}, \quad W_i^K \in \mathbb{R}^{d_{model} * d_k}, \quad W_i^V \in \mathbb{R}^{d_{model} * d_v}$$

and

$$W^O \in \mathbb{R}^{hd_v * d_{model}}.$$

The Transformer, the first sequence transduction model based entirely on attention, replaces the recurrent layers and is most commonly used in encoder-decoder architectures with multi-headed self-attention.

3.2. Neighborhood-based collaborative filtering with attention

One of the first versions of attention applications in recommender systems was implemented for the neighborhood-based model by Mingsheng Fu and Hong Qu [9]. The key difference between neighborhood-based and model-based collaborative filtering systems is that model-based approach utilizes items similarity in combination with users similarity for rating prediction, however, neighborhood-based models use the only items similarity or users similarity, but not their combination. For prediction of the rating $\hat{r}_{u,i}$, neighborhood-based attention model uses a weighted average of known ratings of user u and the weight for a certainly known rating $r_{u,j}$ which is the value of attention between items i and j:

$$\hat{r}_{u,i} = \sum_{j \in N^k(u)} a_{i,j} r_{u,j},$$

where $N^k(u)$ is the items of the user u, and $a_{i,j}$ is the attention value of the item j in relation to the target item i. Embedding vectors of the items i and j are randomly initialized with Gaussian distribution. Attention values are calculated using the following formula:

$$a_{i,j} = Softmax(\exp e_i^T \hat{e_j}).$$

Embeddings e_i and $\hat{e_j}$ are learned by minimizing the difference between $\hat{r}_{u,i}$ and $r_{u,i}$.

4. Proposed method

The interactions matrix can be represented as a bipartite graph (Fig. 1).

Graph Neural Networks aim to generalize neural networks to non-Euclidean domains such as graphs and manifolds. GNNs iteratively build representations of graphs through recursive neighborhood aggregation (or message passing), where each graph node gathers features from its neighbors to represent the local graph structure. According to [10], transformers can be regarded as GNNs which use self-attention for neighborhood aggregation on fully-connected node graphs.

We represent a user through the items, this user interacted with, analogically we represent the item — through the users, who interacted with this item. Basically, we represent the graph node through its neighbors.

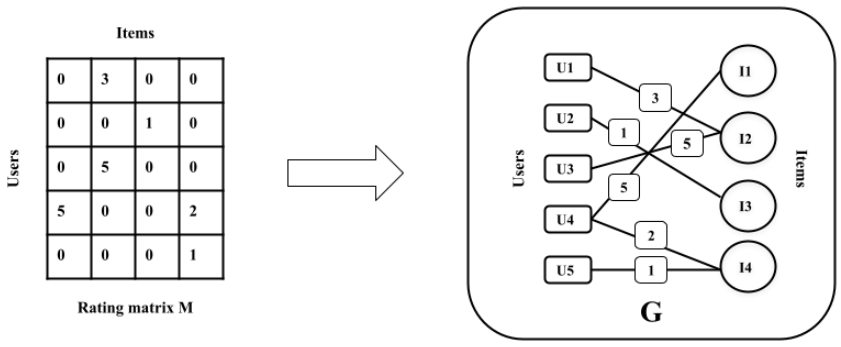

Fig. 1. User-Item interactions matrix representation as a bipartite graph

Let us model the interaction between User 4 (U_4) and Item 4 (I_4). First, we have to get U_4 representation. U_4 has 2 neighbors I_1 and I_4 (Fig. 2).

Inspired by NLP where each transformer input is a fully connected graph, which is one sentence, we will define U_4 representation as a sequence: $U_4 = \{I_1, I_4\}$. We do not include U_4 in the sequence intentionally, because users and items have different modality and embeddings should be optimized separately. These experiments we will leave for further work.

Now let us define the representation of the I_4. This item has interactions with U_4 and U_5 (Fig. 3).

So we got the representation of I_4 as a sequence $I_4 = \{U_4, U_5\}$. Now the goal is to approximate the function $f(U_i, I_j) = r_{ij}$. For $i = 4$ and $j = 4$, it will be equivalent to $f(\{I_1, I_4\}, \{U_4, U_5\}) = 2$.

We define a transformer input sample as a pair of two sequences: sequence of the item (all neighbor users of this item) and sequence of the user (all neighbor items of this user). The target will be to predict the rating. For better convergence of the regression model, we will normalize the target to the scale of $[0, 1]$ using $min - max$ transformation. The proposed method compared to the work in the paper [9] is not neighborhood-based. It is a model-based collaborative filtering system, because at each rating $\hat{r}_{u,i}$ prediction step, it takes into account user's u interactions history and item's i users associated. In the experiments, we will compare the proposed method with different baselines, including traditional methods and a novel method, based on the attention mechanism.

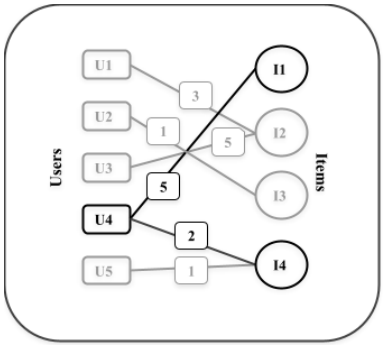

Fig. 2. User 4 graph representation

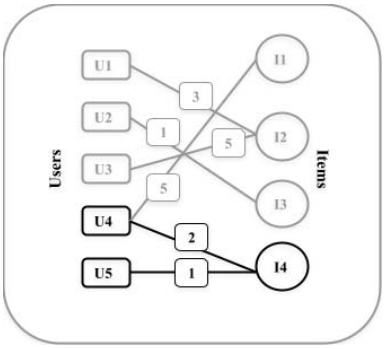

Fig. 3. Item 4 graph representation

5. Experiments

For the experiments, we used a publicly available MovieLens dataset [11]. To investigate the dependence of performance on the amount of data we used 100K, 1M, 10M versions of the dataset.

We compared our method with the following baseline methods.

- **Alternative least squares**. The iterative method of matrix factorization [4]. We used *implicit.als.AlternatingLeastSquares* framework implementation of the algorithm. We tuned model hyperparameters using a random search method.
- **LightFM**. Another implementation of matrix factorization. In contrast to the ALS algorithm, LightFM uses different optimization techniques, based on *"Adagrad"* or *"Adadelta"* optimizers. It is not an iterative method. The framework includes different versions of loss functions: *"bpr"*, *"warp"*, *"warp-kos"* [5].
- **Matrix factorization based on SGD**. We implemented vanilla matrix factorization [4]. This implementation uses stochastic gradient descent as an optimization algorithm.
- **Neural collaborative filtering**. As a neural network-based approach we used neural collaborative filtering with *L2* regularization and *"Adam"* optimizer [7].
- **Neighborhood-based CF with attention**. The architecture is currently reported as a state-of-the-art model [9].

For evaluation, we used *Root Mean Square Error (RMSE)* on a global random 90:10 split, which can be computed as follows:

$$RMSE(\hat{r}) = \sqrt{\frac{\sum_i \sum_j I_{ij}(r_{ij} - \hat{r}_{ij})^2}{\sum_i \sum_j I_{ij}}},$$

where I_{ij} indicates that entry (i, j) appears in the test set. Summarised experiments results are presented in the Table.

Despite the proposed method not outperforming current state-of-the-art methods, it still shows competitive scores on medium size datasets and the best performance on the huge dataset compared to other reproduced methods. It is worth pointing out that we could not reproduce the scores of the neighborhood-based attention model from the original paper, because it is not publicly available yet. In the Table we used original evaluation scores from the paper [9]. We will continue experiments with transformer-based collaborative filtering. Because the training process is computationally expensive, we are limited in hyperparameter tuning. For further experiments we will customize the transformer model to achieve the best performance on MovieLens datasets.

Table

Algorithms comparison on public dataset MovieLens

Method	MovieLens 100K	MovieLens 1M	MovieLens 10M
ALS	1.72	1.36	1.45
LightFM	3.23	2.82	3.07
Vanila MF	0.95	0.92	1.07
Neural CF	0.878	0.91	0.88
Neighborhood attention CF	–	0.836	0.766
Transformer	1.04	0.91	0.83

Conclusion

In this paper, we proposed an approach of training a state-of-the-art NLP technique to address a collaborative filtering task. We have shown how to achieve competitive results on the basic RecSys MovieLens dataset. We believe that in any machine learning task it can be beneficial to look for ideas and inspiration in different machine learning domains, like in our case in natural language processing. We have to note that reported results are suboptimal because as pointed out in the paper [12], the correct model evaluation requires significant effort on hyperparameters tuning and experiments setup.

References

1. Gomes-Uribe C. A., Hant N. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Transactions on Management Information Systems*, 2016, vol. 6, no. 4, pp. 1–19. https://doi.org/10.1145/2843948
2. Liu S., Bismas P. K. A Hybrid Recommender System for Recommending Smartphones to Prospective Customers. *CoRR*, 2021, vol. 23, pp. 2–3.
3. Strömqvist Z. *Matrix factorization in recommender systems. How sensitive are matrix factorization models to sparsity*? Department of Statistics Uppsala University, 2018. 26 p.
4. Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, Tat-Seng Chua. Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, vol. 10, pp. 335–344. https://doi.org/10.1145/3077136.3080797
5. Wang K., Peng H., Jin Y. Sha Ch., Wang X. Local Weighted Matrix Factorization for Top-n Recommendation with Implicit Feedback. *Data Science and Engineering*, 2016, vol. 1, pp. 252–264. https://doi.org/10.1007/s41019-017-0032-6
6. Rendle S., Freudenthaler C., Gantner Z., Schmidt-Thieme L. Bayesian Personalized Ranking from Implicit Feedback. *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009, vol. 10, pp. 452–461.
7. Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, Tat-Seng Chua. Neural Collaborative Filtering. *Proceedings of the 26th International Conference on World Wide Web*, 2017, vol. 10, pp. 173–182. https://doi.org/10.1145/3038912.3052569
8. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I. Attention is all you need. *Conference on Neural Information Processing Systems*, 2017, vol. 15, pp. 4–5.
9. Mingsheng F., Hong Q., Dagmawi M., Li L. Attention based collaborative filtering. *Neurocomputing*, 2018, vol. 311, pp. 88–98. https://doi.org/10.1016/j.neucom.2018.05.049
10. Veličković P., Casanova A., Lio P., Cucurull G., Romero A., Bengio, Y. Graph attention networks. *6th International Conference on Learning Representations, ICLR 2018 – Conference Track Proceedings*, 2018, vol. 12, pp. 2–3. https://doi.org/10.17863/CAM.48429
11. Harper F. M., Konstan J. A. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems*, 2015, vol. 5, iss. 4, pp. 1–19. https://doi.org/10.1145/2827872
12. Rendle S., Zhang L., Koren Y. On the Difficulty of Evaluating Baselines. *ArXiv preprint arXiv:1905.01395*, 2019, vol. 19, pp. 1–3.

Поступила в редакцию / Received 24.11.2021
Принята к публикации / Accepted 21.12.2021
Опубликована / Published 31.03.2022