applied sciences

MDPI

Article

Sequential Recommendations on GitHub Repository

JaeWon Kim, JeongA Wi and YoungBin Kim *

Department of Image Science and Arts, Chung-Ang University, Dongjak, Seoul 06974, Korea; jaewon.k92@gmail.com (J.K.); placeja@cau.ac.kr (J.W.)
* Correspondence: ybkim85@cau.ac.kr

Abstract: The software development platform is an increasingly expanding industry. It is growing steadily due to the active research and sharing of artificial intelligence and deep learning. Further, predicting users' propensity in this huge community and recommending a new repository is beneficial for researchers and users. Despite this, only a few researches have been done on the recommendation system of such platforms. In this study, we propose a method to model extensive user data of an online community with a deep learning-based recommendation system. This study shows that a new repository can be effectively recommended based on the accumulated big data from the user. Moreover, this study is the first study of the sequential recommendation system that provides a new dataset of a software development platform, which is as large as the prevailing datasets. The experiments show that the proposed dataset can be practiced in various recommendation tasks.

Keywords: dataset; deep neural network; implicit feedback; recommendation system; sequential recommendation systems

Citation: Kim, J.; Wi, J.; Kim, Y. Sequential Recommendations on GitHub Repository. *Appl. Sci.* **2021**, *11*, 1585. https://doi.org/10.3390/app11041585

Academic Editor: Ángel González-Prieto and Fernando Ortega
Received: 14 December 2020
Accepted: 2 February 2021
Published: 10 February 2021

1. Introduction

GitHub is one of the biggest software development platforms, wherein a large number of users upload their open-source projects. It has gained popularity at an expanding rate, as depicted in Figure 1. It stores a vast number of repositories of source codes related to researchers' projects or research papers for them to share with more people online. The total number of repositories reached more than 280 million, and the total number of users on this platform reached 69 million users in this platform in August 2020 (GitHub API https://api.github.com/) (accessed on 20 Nov 2020). As a result of the augmentative computation-intensive nature of modern scientific discovery, this trend is largely growing in deep learning (DL) and machine learning fields; a particularly clear example can be found where links are often provided to GitHub in published research papers. However, to find useful information or repositories in GitHub, one needs to inspect projects manually using a search filter or search for a popular project on the Explore GitHub page (https://github.com/explore). Subsequently, the page provides you with categories that are limited to static themes such as 'recently visited repositories', 'recently visited topics', 'starred repositories', or trending projects in consideration of the recently visited topics. GitHub provides such recommendations based on general recommendation topics, although it depends heavily on temporally close and content-based relationships (category, language, etc.) as far as we have experienced.

The recommendation system is an algorithm that proposes an item related to or preferred by the user. Effective recommendations have become essential as recommenders are applied in a large number of fields, and their contents increase at an exponential rate. Consequently, many studies have been conducted on this subject. The emergence of deep learning has contributed to a significant improvement in the research. Various models and techniques have been proposed, including neural collaborative filtering (NCF) [1], neural factorization machine [2,3], recurrent neural network (RNN) [4–7], convolutional neural network (CNN) [8,9], and reinforcement learning models [10]. All models have a tendency

to recommend items based on different points of interest depending on their particular task. Some systems handle user-based settings, while some deal with item-based settings regarding the user's general long preferences, personalizations, or sequential interactions.

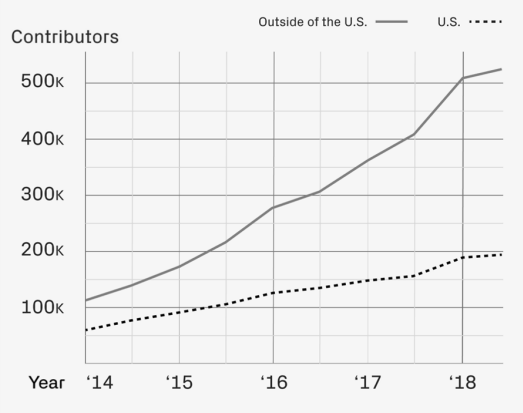


Figure 1. Rising popularity in GitHub from 2014 to 2018. (https://github.blog/2018-11-08-100m-repos/) (accessed on 18 December 2020).

Most traditional recommendation systems are content-based and collaborative filtering (CF)-based systems. They tend to model their preferences for items based on explicit or implicit interactions between users and items. Specifically, they assume that all user–item interactions in historical order are equally important, and try to learn one's static preferences using the users' history of interactions. However, this may not be maintained in real-life scenarios, wherein the users' next actions rely heavily on their current intentions as well as static long-term preferences, which may be deduced and influenced by a small set of the most recent interactions. Further, conventional approaches ignore sequential dependencies between user interactions, and user preferences are modeled incorrectly. Consequently, sequential recommendations are gaining popularity in academic research and practical applications. Prevailing recent recommendation systems, such as the convolutional sequence embedding recommendation model (Caser) [8] and self-attentive sequential recommendation (SASRec) [11] utilize interaction sequences that contain useful information about each user's behavior, e.g., listening to music, purchasing merchandise, watching YouTube, and the similarity between the items, by capturing both long-range and short-range dependencies of user–item interaction sequences.

In this study, we tested a recommendation algorithm using GitHub interaction datasets to determine whether it can adequately obtain sequential interactions regarding academic and research interests in large datasets. We evaluated our dataset by implementing gated recurrent units for recommendation (GRU4Rec) [4], Caser, and SASRec methods, which are DNN-based sequential recommendation algorithms. Based on sequential recommendation studies [8,9,11,12], we assume that it is sufficient to test the recommendation in this field by supplying the model with only the user's item preference sequence. In a recommendation system task, cold start refers to the situation when there is not enough recorded data to recommend any items to new users. In summary, when cold starters are people with less than 40 interactions and repositories with less than 160 interactions, normalized discounted cumulative gain [13] (NDCG) scores of 0.067 and 0.145 were achieved for ten items, along with precision scores of 0.013 and 0.018. The contributions of this research are as follows.

- As far as we are aware, we are the first to provide a large-scaled GitHub dataset for a recommendation system. We provide two different scales of the dataset, which contains approximately 19 million interactions with over 230,000 users and 102,000 repositories.
- We present an in-depth experiment on recommendations with the GitHub dataset.
- We introduce the potential of sequential recommendations in the researchers' platform.

2. Related Works

In this section, we describe the related works based on three aspects: recommendation system studies in GitHub, general recommendation systems, and sequential recommendation systems.

2.1. Recommendations in GitHub

Millions of repositories and users have greatly facilitated researchers' studies; however, only a number of studies have focused on this particular platform [14–22]. One of the pioneering works [15,19,20] based on term frequency-inverse document frequency, is usually used to reflect the importance of a word to a document in a collection or corpus [23]. Furthermore, Zhang et al. and Shao et al. [16,22] are the most recent and the only DNN-based recommendation studies in this specific field that utilize deep auto-encoder and graph convolutional networks (GCNs). Zhang et al. [22] introduce Funk singular value decomposition Recommendation using Pearson correlation coefficient and Deep Auto-Encoders (FunkR-pDAE) which applies modified singular value decomposition to optimize similarity of user and item matrices and deep auto-encoder to learn the latent sparse user and item features. Shao et al. [16] encode both the graph neighborhood and content information of created nodes to be recommended or classified [24], to capture interactions between users and items. Despite the fact that deep learning-based recommendation systems are superior to traditional item-based or machine learning-based approaches, the studies are based on the content; and they are not able to represent time or sequential interaction patterns.

2.2. General Recommendation Systems

Generally, the recommendation system recommends an item to a user, which another user with the similar preferences was interested in, based on the interaction history [25–27] using CF. Although such traditional machine learning-based algorithms are replaced by modern deep learning-based algorithms, they pioneered this field of study. To be specific, non-negative matrix factorization is a method [26] that projects items and users into latent spaces through the exploitation of global information with internal products of vectors and predicts the user's interest in items. The two-layer Restricted Boltzmann Machine [28] method is another popular algorithm in CF. Salakhutdinov [28] pioneered DNN-based algorithms and won the Netflix Prize (https://www.netflixprize.com). NFC [1], AutoRec [29], and Collaborative Denoising Auto-Encoder [30], using auto-encoders along the lines of DNNs, have replaced the existing algorithms; however, the principle cause for the decline of these systems is that they cannot represent sequential information.

2.3. Sequential Recommendation Systems

As GitHub provides recent and various fields of projects, researchers tend to refer to projects depending on their current interests, e.g., projects or studies. Therefore, temporal and sequential interactions in GitHub are essential information for a recommendation system. The sequential recommendation system mainly aims to forecast and predict consecutive items based on the user's past sequential interaction patterns in chronological order. The growing popularity of these systems is due to the robustness of the cold-start problem; moreover, there is no ignorant information within the user's sequential behavior (e.g., purchasing item, listening to music, and streaming videos). The initial approach was the Markov chain-based model, Rendel et al., He et al., and Zhang et al. [31–33], which recommended based on L previous interactions with the L-order Markov chain. The next paradigm was the RNN-based models, such as the GRU4Rec [4] or RNN model [5,7], that were used to denote the representation in the user behavior sequence. This approach has attracted attention because RNNs are proficient in modeling sequential data [34]. In contrast, CNN-based models have been proposed by Tang and Wang [8] using horizontal and vertical convolution filters to address the vulnerability of RNNs in gradient vanishing problems, while modeling sequences as an option to learn sequential patterns.

Recently, several studies have attempted to utilize neural attention mechanisms for using sequential interactions and improving recommendation performance [35–37]. Attention mechanisms learn to focus only on the features that are important to a given input. Therefore, it has been studied actively in natural language processing (NLP) and computer vision. In addition, unlike the standard algorithms, the self-attention mechanism [38] models a complex sentence structure and searches for related words by generating the next word by considering the relationship between the word and other words. Various approaches using the attention mechanism [11,12,39] have been proposed to represent the sequential behavior of users through high-level semantic combinations of elements, and have achieved cutting-edge results. Consequently, we have adopted the sequential recommendation models to evaluate the recommenders in modeling extensive user data of an online community.

3. Dataset and Models

3.1. GitHub Dataset

In this section, the dataset obtained from the GitHub database will be discussed beforehand. The data were obtained and crawled from the public GitHub Archive (https://www.gharchive.org/) (accessed date 18 December 2020). Although activity archives are available from 12 February 2011 to the present day, we have selected data only from the year of 2018. The database includes the repository id, hashed user id, program language, and description of each repository. Based on this online database, we constructed the GitHub dataset, as shown in Table 1. The dataset is formed in the Python dictionary format under the pickle module. We assumed that the interaction between a user and item is created when he or she stars a repository. This speculation was made because the users' repository visitation logs were not available. They may repetitively or simultaneously visit multiple repositories; however, the information regarding the track of this action is publicly not accessible, so it is limited to users' own will of starring the repositories. Additionally, as the timestamp of each activity could not be retrieved and under the assumption that the exact time information is not a necessary feature, we have excluded this feature in our GitHub dataset.

The fundamental information of each feature will be explained further due to the pre-processing of the data for efficiency. First, each dataset under different numbers of cold-start interactions is mainly divided into train, validation, and test data in the Python dictionary format. This will be elaborated in more detail in the next section. Second, within that dictionary, each repository, hashed user index, and program language is given a unique id. Finally, the raw description of a repository is provided with the parsed vocabulary and their unique id for anyone interested in content-based recommendation. The GitHub link (https://www.github.com/John-K92/Recommendation-Systems-for-GitHub) to download our dataset and a custom dataset pre-processing method are provided for readers to utilize this dataset on any DL models. This custom dataset pre-processing method is based on Pytorch Spotlight Interaction module which is widely used for recommendation tasks with Pytorch framework to handle interaction sequences of user and item actions. You may refer to spotlight documentation [40] for more details. Other popular datasets of recommendation studies such as, MovieLens, and Amazon (http://jmcauley.ucsd.edu/data/amazon/), are compared based on the statistics from each dataset in Table 2. For every dataset, the interactions were assigned by the implicit feedback (review or ratings of items) of each user and item. As seen in Table 2, the domains and scarcity of real-world datasets differ significantly. It can be clearly noticed that our GitHub dataset is the largest amongst all the well-known datasets. The size of the dataset that is too large may be considered as having a lot of items rather than the user–item interaction information which is a powerful signal in the recommendation system. However, when the interaction cold-start is set as 2, the number of users, number of items, and total log are reduced to 2,572,450, 2,054,002, and 55,380,271, respectively. As a result, the GitHub dataset was utilized by adjusting the appropriate cold-start number in our experiment. The GitHub dataset is an interesting

database for researching and developing recommendation systems, given their size, relative scarcity, and average interaction.

Table 1. Summary of GitHub dataset features.

GitHub Dataset		
Feature	**Type**	**Description**
train	Interactions dataset	Note to refer to a custom spotlight file to explore each sequence of train dataset
valid	Interactions dataset	Note to refer to a custom spotlight file to explore each sequence of valid dataset
test	Interactions dataset	Note to refer to a custom spotlight file to explore each sequence of test dataset
repo_index2id	list	Even though it is in a list format it is used to retrieve unique ids within a custom spotlight file
user_index2id	list	Even though it is in a list format it is used to retrieve unique ids within a custom spotlight file
languages	list	A unique set of computer program languages list in all repositories
lang2id	dict	A python dictionary format with each language and its id
id2lang	dict	A python dictionary format with each repository id and its language id
descriptions	dict	A python dictionary format with each repository id and its text description
descriptions_parsed	dict	A python dictionary format with each repository id and list of its parse description ids
voca	list	A list of parsed vocabulary from repository descriptions
voca2id	dict	A python dictionary formation with parsed vocabulary and its unique id

Table 2. Statistics of datasets.

Datasets	**#Users**	**#Items**	**#Interactions**	**Avg.inter./User**	**Avg.inter./Item**	**Sparsity**
ML 1M	6,022	3,043	999,154	165.25	327.03	94.57%
ML 20M	138,493	18,345	19,984,024	144.30	1089.34	99.21%
Beauty	22,363	12,101	198,502	8.88	16.40	99.93%
Video Games	108,063	31,390	21,105,584	9.54	21.72	99.38%
BGG [37]	388,413	87,208	47,351,708	121.91	542.97	99.86%
GitHub	4,135,902	5,264,307	59,765,590	14.45	11.35	99.99%

3.2. Baseline Recommendation Models

In our experiment, we assume a set of given users as $\mathcal{U} = \{u_1, u_2, ..., u_{|\mathcal{U}|}\}$ and a set of repositories as $\mathcal{R} = \{r_1, r_2, ..., r_{|\mathcal{R}|}\}$. $S^u = \{r_1^u, ..., r_t^u, ..., r_{|S^u|}^u\}$ represents the interaction sequence in chronological and sequential order of each user, where user u interacts at time step t under $r_t^u \in \mathcal{R}$. However, unlike other tasks, t stands for the sequential order of interactions, not the absolute timestamp as in temporal-based recommendation systems [5,41,42], similar to the prominent DL-based recommendation models, Caser, BERT4Rec [12], and sequential-based methods [6,43].

3.2.1. GRU4Rec

The GRU4Rec [44] model focuses on modeling model interaction sequences in session-based scenarios. The backbone structure of GRU4Rec uses a variant of RNN, gated recurrent units (GRUs), which is a more sophisticated version of RNN that effectively alleviates the vanishing gradient problem of an RNN with an update gate and a reset gate. The gates in GRU mainly learn to control the amount of information that is required to update and forget the hidden state of the unit. For instance, a reset gate allows control of how much information to remember from the previous state. Similarly, an update gate allows control of how much of the new state is a copy of the old states. Considering $\mathbf{W}$ and $\mathbf{U}$ as the learned weight at each gate and the information of the hidden gate as h, the update gate is calculated with input x into the network at a time step t,

$$z_t = \sigma(\mathbf{W}_z x_t + \mathbf{U}_z h_{t-1}), \tag{1}$$

and the reset gate is determined by

$$r_t = \sigma(\mathbf{W}_r x_t + \mathbf{U}_r h_{t-1}), \tag{2}$$

where sigmoid $\sigma(\cdot)$ is an activation function for the non-linear transformation. The memory content that stores information from the previous gate using the reset gate is given as follows:

$$\hat{h}_t = tanh(\mathbf{W} x_t + r_t \odot \mathbf{U} h_{t-1}), \tag{3}$$

where $\odot$ is an element-wise product of elements. The determination of what to store in the current memory content within the past and current steps by final linear interpolation is determined as

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \hat{h}_t. \tag{4}$$

The GRU4Rec outperforms the RNN-based algorithms; the basic GRU4Rec's architecture is shown in Figure 2. In this illustration, L represents the previous items and T represents the target item sequences within the user interaction sequence Su, respectively, and $\mathbf{E}$ represents the embedding matrix of the L previous consecutive interactions. To capture inter-dependencies, Hidashi et al. [44] further improved recommendation accuracy by introducing new ideas, such as TOP1 losses [44], Bayesian Personalized Ranking loss [45], and mini-batch negative sampling.

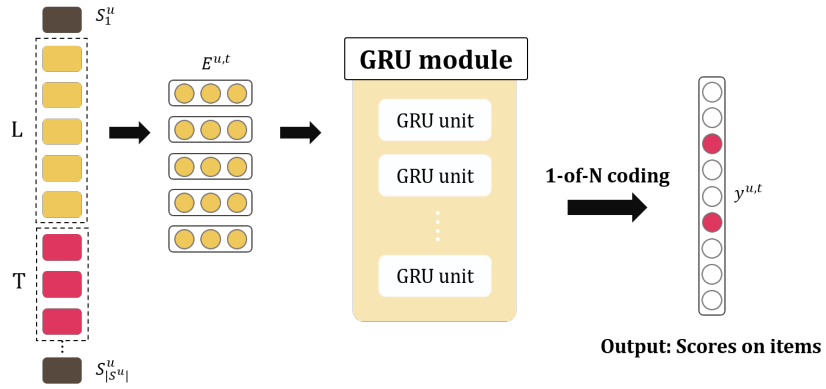


Figure 2. Model architecture of gated recurrent units for recommendation (GRU4Rec) [44].

3.2.2. Caser

In order to learn the sequential representations from user and repository interactions, this model applies both horizontal and vertical convolutional filters to leverage a latent factor for the CNN. The convolution processing treats an embedded matrix $\mathbf{E} \in \mathbb{R}^{\mathcal{L} \times d}$ as an input image, when $\mathcal{L}$ is the historical interacted items in dimensional latent space d [8]. Caser [8] learns the sequential pattern of interactions by feeding the local features of the image and the general preferences of users. The horizontal convolution estimates

$\phi_c(\mathbf{E}_{i:i+h-1} \odot \mathbf{F}^k)$, given that ϕ_c is an activation function in convolution layers and h is the height of the filter, as illustrated in Figure 3. Subsequently, the vertical convolution captures the latent features by $\sum^{\mathcal{L}_{i=1}} \tilde{\mathbf{F}}_l^k \cdot \mathbf{E}_l$, with the fixed filter size at $1 \times L$ unlike the horizontal convolution layer. The two stream outputs of these two convolution layers are then fed into a fully connected layer to attain abstract and higher-level features. Finally, the output $\mathbf{z}$ and the user's embedded general preference $\mathbf{P}^u$ are concatenated and the result is then linearly projected to an output layer, which is written as

$$y^{(u,t)} = \mathbf{W}' \begin{bmatrix} \mathbf{z} \\ \mathbf{P}^u \end{bmatrix} + \mathbf{b}', \tag{5}$$

where $\mathbf{W}' \in \mathbb{R}^{\mathcal{J} \times 2d}$ and $\mathbf{b}' \in \mathbb{R}^{\mathcal{J}}$ with nodes $|\mathcal{J}|$ are the weight matrix and bias term in the output layer, respectively.

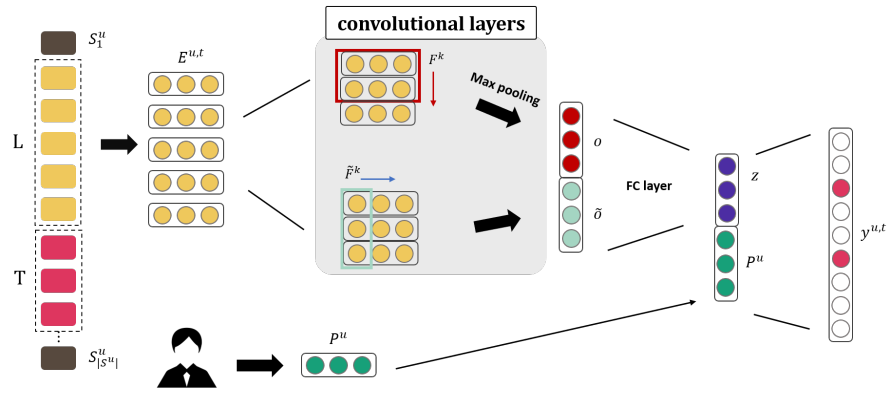


Figure 3. Model architecture of Caser [8].

3.2.3. SASRec

Kang [11] made use of a self-attention module, comprising a left-to-right unidirectional single-head attention model, to capture interaction representation. The architecture of the model is elaborated in Figure 4. This means that it identifies relevant or essential items from historical sequential interactions and forecast the next item in sequence. Unlike CNN- or RNN-based models, it is cost-efficient and is able to model semantic and syntactic patterns among interactions. SASRec tends to focus on long-term dependence in dense datasets, while simultaneously focusing on relatively recent interactions in sparse datasets. The attention of the scaled dot product can be obtained by

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = Softmax(\frac{\mathbf{Q}\mathbf{K}^\intercal}{\sqrt{d}})\mathbf{V}, \tag{6}$$

where the inputs, the query, keys, and values, that are fed into the self-attention block are denoted as $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, respectively. Here, d is the dimensionality, and $\sqrt{d}$ denotes the scale factor that helps to avoid the unwilling large values of the inner product in the output of the weighted sum of values from the interactions between the query and key. Each self-attention head is calculated as follows:

$$\mathbf{A} = Attention(\mathbf{E}\mathbf{W}^Q, \mathbf{E}\mathbf{W}^K, \mathbf{E}\mathbf{W}^V). \tag{7}$$

Then, the output is fed to a point-wise two-layer feed-forward network sharing the parameters. Thus, the self-attention block can be re-written as:

$$\mathbf{F}_i = FFN(\mathbf{A}_i) = ReLU(\mathbf{A}_i\mathbf{W}^1 + b^1)\mathbf{W}^2 + b^2. \tag{8}$$

Therefore, output $\mathbf{F}_i$ captures the all user-item sequences. However, Kang et al. [11] also maintained the stacking of another self-attention block to learn more complex interactions from the sequential item information.

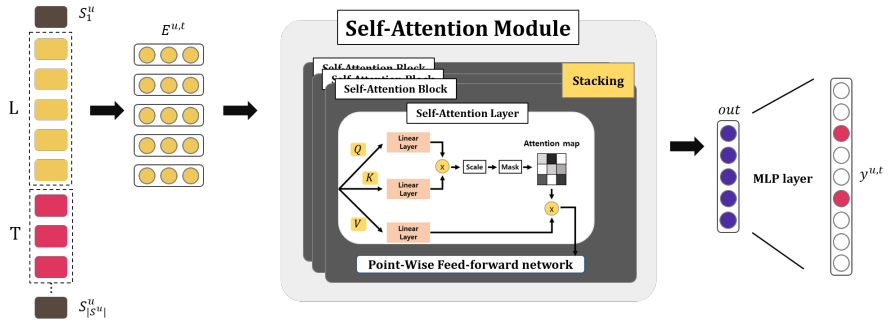


Figure 4. Model architecture of self-attentive sequential recommendation (SASRec) [11].

4. Experiments

4.1. Experimental Setup

4.1.1. Dataset

This section first describes the pre-processed GitHub dataset set with statistics and evaluates the recommender models and their variations. We then elaborate on the results and analyses of the experiments of the recommendation. For a reasonable comparison, the learning and evaluation code was imported from each appropriate author.

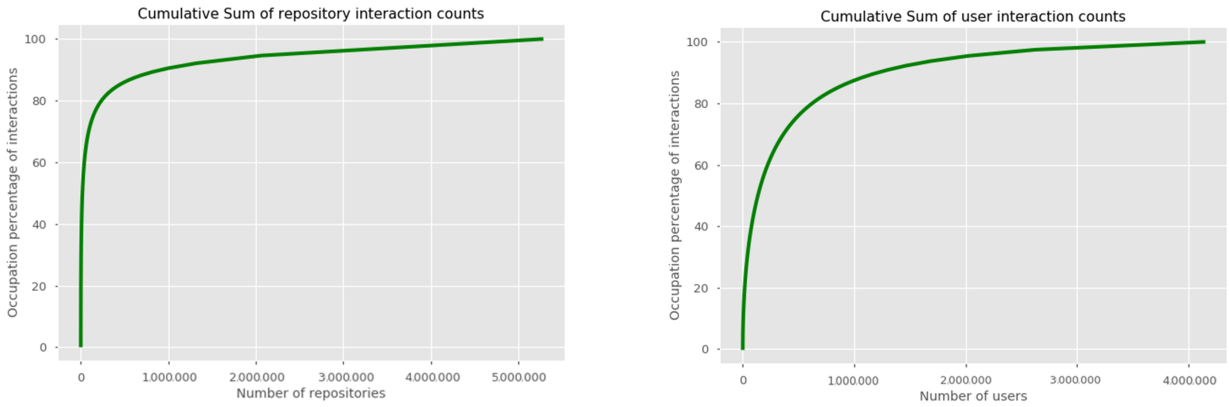
As mentioned in the Section 3.1, we focus on the interactions between user and repository. Therefore, although a repository has attributes such as timestamp, description, and program language, we exclusively focus on the interactions. In that respect, we ruled out the cold-start users and repositories under n number of interactions, where n was set to 40 and 160. The reason behind this is that sparsely interacted items may act as outliers within the dataset and to ensure strong relations among the interactions as practiced in common recommendation systems in [1,8,32,43,46]. The size n of cold-start in the GitHub dataset indicates that there are no items or users that interact with less than the n number of inter-activities. Along with n cold-start, we set the L previous interactions as 5 and that for target sequence T as 3. For details, Table 3 shows a brief analysis of the two datasets for our training and experiments. The total number of interactions is revised from the original 37,068,153 interactions to 19,064,945 and 2,968,165, in respect to the n values. n can be varied; however, we have set the value to specific numbers to acquire adequate correlativity for modeling recommendation tasks in consideration of the data sparsity. As the sparsity of a dataset significantly affects the modeling of the recommendation, we have defined n as two values to vary the sparsity. Consequently, the number has dropped from 37 million to a minimum of approximately 2 million, but the amount is still significant for evaluating the recommendation performance. In this dataset, data were split for testing and validation with 20% and 10% of the dataset for each n-length.

Table 3. Summary of statistics of preprocessed datasets.

Descriptive Statistics		
minimum number of interactions	40	160
number of interactions	19,064,945	2,968,165
number of unique users	230,469	14,213
number of unique repository	102,004	8,645
data sparsity	99.92%	97.58%
average interactions per user	82.72	208.83
average interactions per repository	196.9	343.34

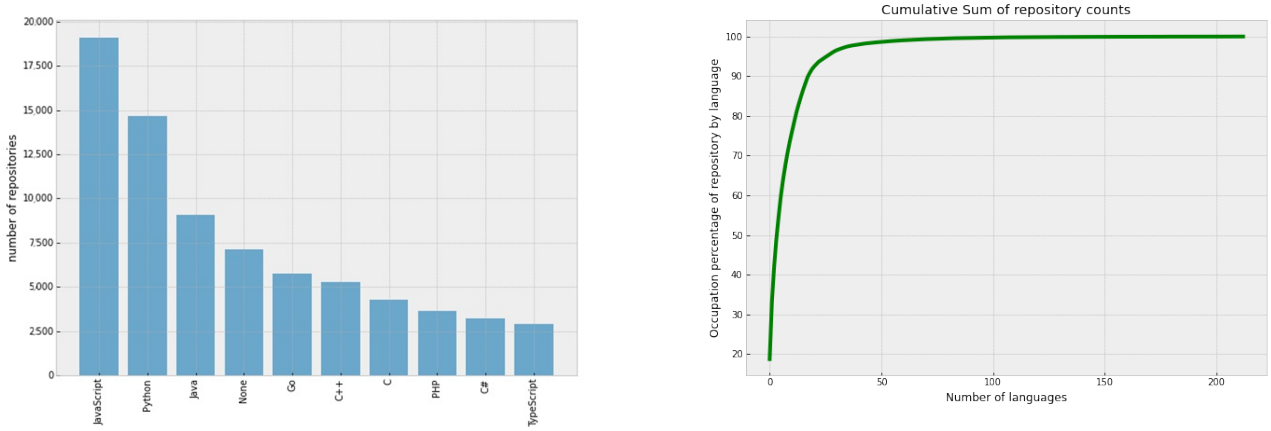
When applying recommendation systems using a certain dataset, setting the appropriate minimum number of historical interactions and often modeling subsidiary information,

such as based language and description, is imperative. First, the GitHub dataset is skewed to only a small number of repositories occupying a significant amount of interactions similar to other recommendation task datasets. As shown in Figure 5a, approximately less than 20% of the most interacted repositories explain almost 90% of all the interactions within the dataset. In comparison, the users are relatively evenly distributed, as shown in Figure 5b. This indicates that the amount of information loss, resulting from setting the value n, is inconsistent. Second, for n size of '40', which is the minimum length in our experiment, Figure 6 represents the brief statistics of the language feature in the dataset. Although we have not used the language, description, or time information which could be used effectively in the recommendation system, and used only sequential interactions, we present the concise statistics of the 'language' feature. Figure 6a illustrates the top 10 most program-based repositories. Here, one of the major features is "None" values, and a few languages represent the most repositories in the dataset, as seen in Figure 6b. Therefore, careful attention is needed for those who are interested in using such 'language' features in the GitHub dataset. However, because of the large number of repositories of common program languages, it can still be a prominent feature in recommendation tasks. The supplement feature handling can help the model performance.



(**a**) Cumulative sum of repository interactions. (**b**) Cumulative sum of user interactions.

Figure 5. Statistics of the number of games and number of users.



(**a**) Number of repositories for each 'language' feature. (**b**) Cumulative sum of repositories by language.

Figure 6. Statistics of the number of repositories per language.

4.1.2. Evaluation Metrics

In the evaluation of the recommendation systems, we implemented various common evaluation metrics in the task, including precision@N, recall@N, mean average precision (MAP), normalized discounted cumulative gain (NDCG@N), and mean reciprocal rank (MRR) as in studies [8,11,12,31,47,48]. Precision@N and Recall@N are determined by

$$\begin{aligned} Prec@N &= \frac{|\mathcal{R} \cap \hat{\mathcal{R}}_{1:N}|}{N}, \\ Recall@N &= \frac{|\mathcal{R} \cap \hat{\mathcal{R}}_{1:N}|}{|\mathcal{R}|}, \end{aligned} \tag{9}$$

where $\hat{\mathcal{R}}_{1:N}$ is the top-N estimated repositories per user. MAP is the mean of the average precision (AP) of all users in $\mathcal{U}$, given $rel(N) = 1$ if the ground truth $\mathcal{R}$ is equal to the predicted Nth item in $\hat{\mathcal{R}}$; AP is computed by

$$AP = \frac{\sum_{N=1}^{|\hat{\mathcal{R}}|} Prec@N * rel(N)}{|\hat{R}|}. \tag{10}$$

NDCG@N implies a position-aware metric that assigns rank-specific weights to measure rank quality. It can be calculated as

$$NDCG@N = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{2^{rel}_{\mathcal{R}_{u,g_u}} - 1}{log_2(\mathcal{R}_{u,g_u} + 1)}, \tag{11}$$

where $\mathcal{R}_{u,g_u}$ is the predicted rank for g_u, when g_u is the ground truth repository for a given user u. In recommendation tasks, MRR refers to how well the recommender model assigns rankings similar to NDCG. To distinguish between the two indicators, MRR focuses on the assessment of assigning ground-truth items to higher ratings in forecasts; however, the NDCG considers rankings in relation to adding larger weights from higher ratings. MRR is defined as follows:

$$MRR = \frac{1}{M} \sum_{u \in \mathcal{U}} \frac{1}{\mathcal{R}_{u,g_u}}. \tag{12}$$

4.2. Performance Analysis

In this experiment, we have set the number of $\mathcal{L}$ as 5 and the number of $\mathcal{T}$ as 3, which heuristically gave the best performance in training all given models. That is, the input number of each item sequence is 5, predicting the next 3 items in a sequence for each user. The numbers of two cold-start datasets are set to 40 and 200, under the assumption that similar data sparsity of a dataset will result in a similar result.

The results of the recommendation algorithms described previously are shown in Table 4. The CNN-based model surpassed all the models in all matrices. Generally, self-attention-based recommendation systems such as SASRec, BERT4Rec, and AttRec [39] better predict the subsequent items in relatively sparse datasets, especially when CNN-based recommenders have a limited receptive field [11] in regards to the size of the dataset. These results show the presence of being more weighted on the short-term sequential interactions in GitHub or similar platforms. Nonetheless, recommendations based on the RNN, such as GRU4Rec, model performed worse than the other algorithms. This is particularly because the model learns the sequence in a step-by-step manner, which suggests that too much concentration in the sequence will degrade the overall performance. Consequently, it is necessary to capture dependencies effectively, not only in the short-term but also in long-term information (e.g., users' general preference) [8,9,11,43]. Considering the different n number of cold-start interactions, Caser outperformed the performance indicators for Pres@10 and MAP in more than 200 interactions. The metrics were extremely close to each other when number of cold start is 40 interactions (n = 40), but the gap was

enlarged because they tend to focus more on users with larger number of interactions. It clearly demonstrates that the CNN-based model, Caser, is robustly capable of capturing short-term sequential interactions, as well as accurately modeling long-term preferences especially in the less receptive areas.

Table 4. Recommendation performance comparison of different models on GitHub dataset.

Models	**n = 40**					**n = 160**				
	Prec@10	**Recall@10**	**MAP**	**MRR**	**NDCG@10**	**Prec@10**	**Recall@10**	**MAP**	**MRR**	**NDCG@10**
GRU4Rec	0.0005	0.0004	0.0005	0.0003	0.0021	0.0102	0.0037	0.0069	0.0023	0.0391
Caser	0.018	0.019	0.014	0.010	0.067	0.043	0.017	0.020	0.009	0.145
SASRec	0.013	0.013	0.010	0.007	0.046	0.018	0.007	0.011	0.004	0.069

Figure 7 presents a predicted output of the predicted repositories from the randomly sampled users in the validation dataset. The five sequential interactions are the input items of the model; and we presented the output predicted items. In this figure, 'category' denotes the hand-crafted feature from each item and the 'language' feature is obtained from each output repository. We manually checked the task or project of the output repositories, and categorized the topic to visualize the example. As the repository was predicted from time step t, it was found that the previous time step $t-1$ item not only affected the forecast next item but also the next nearby time steps in the sequential recommender. The highest-ranked $item_{t,r}$ includes a feature ('category') of the immediate previous item and the one prior('language') inclusively. However, $item_{t,r-1}$ item is rather much closer to $item_{t-3}$ and $item_{t-4}$ than the sequentially nearby item, such as second previously interacted item. This reveals that the outputs include various features of sequence perspective and that the recommender can represent users' sequential and general preferences. Figure 7 briefly shows the notion of deep learning-based sequential recommendation and the correlation of typical class features ('type' and 'category') in GitHub recommendation.

Input items

Input repo. sequences	$item_{t-5}$	$item_{t-4}$	$item_{t-3}$	$item_{t-2}$	$item_{t-1}$
"language"	Python	Python	Python	Jupyter notebook	JavaScript
"category"	Voice Processing	Voice Synthesis	Voice Synthesis	Voice Processing	Voice Visualizer

Recommendation System

Predicted items

Predicted repos. in rank	$item_{t,r}$	$item_{t,r-1}$	$item_{t,r-2}$	$item_{t,r-3}$	$item_{t,r-4}$	$item_{t,r-5}$	$item_{t,r-6}$	$item_{t,r-7}$	$item_{t,r-8}$	$item_{t,r-9}$
"language"	Python	Python	Python	Python	JavaScript	Python	Python	Jupyter notebook	Python	JavaScript
"category"	Voice Visualizer	Voice Synthesis	Voice Synthesis	Voice Processing	Voice Visualizer	Voice Synthesis	Voice Recognition	Voice Processing	Voice Synthesis	Voice Synthesis

Figure 7. Example of predicted items.

To sum up, the GitHub platform has an exclusive differentiation between various platforms in recommendation tasks.

5. Conclusions and Future Work

This study presents a DNN-based recommendation system using a large-scale dataset associated with GitHub. The pervasive growth rate and distinct aspects of the use of the GitHub platform suggest that more studies may be carried out in this particular area. Consequently, we investigated the GitHub dataset using prevailing deep learning-based recommendation systems. The recommenders described in the study have been evaluated on GitHub platforms, along with past user–repository interaction sessions. The results shows that the CNN-based approach, Caser, is consistently better than the other recommendation systems algorithms and models personalized preferences and interactions on these platforms regardless of the size of cold-start interactions.

Future studies, therefore, should consider various state-of-the-art recommendation systems to study the properties of GitHub interactions further. These include RNN-based VLaReT [49] that outperformed the mentioned baseline models, and improved CNN-based recommender [50] and hypergraph CNN-based recommender [51]. Moreover, using multiple labeled features, such as language types and NLP-based features by modeling with a pre-trained language model, may be essential for analyzing user interactions and predicting future behavior.

Author Contributions: Conceptualization, Y.K. and J.K.; methodology, J.K. and J.W.; software, J.K.; validation, J.K. and J.W.; data curation, J.K.; writing—original draft preparation, J.K.; writing—review and editing, Y.K., J.K. and J.W.; visualization, J.K.; supervision, Y.K.; project administration, Y.K. and J.K.; funding acquisition, Y.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Chung-Ang University Research Grants in 2020 and by the National Research Foundation of Korea (NRF) through the Korean Government (MSIT) under Grant NRF-2018R1C1B5046461.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available in a publicly accessible repository that does not issue DOIs Publicly available datasets were analyzed in this study. This data can be found here: https://www.github.com/John-K92/Recommendation-Systems-for-GitHub.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.S. Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web, Parth, Australia, 3–7 April 2017; pp. 173–182.
2. He, X.; Chua, T.S. Neural factorization machines for sparse predictive analytics. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 355–364.
3. Guo, H.; Tang, R.; Ye, Y.; Li, Z.; He, X. DeepFM: a factorization-machine based neural network for CTR prediction. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, Melbourne, Australia, 19–25 August 2017; pp.1725–1731.
4. Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-based recommendations with recurrent neural networks. *arXiv* **2015**, arXiv:1511.06939.
5. Wu, C.Y.; Ahmed, A.; Beutel, A.; Smola, A.J.; Jing, H. Recurrent recommender networks. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, Cambridge, UK, 6–10 February 2017; pp. 495–503.
6. Xu, C.; Zhao, P.; Liu, Y.; Xu, J.; Sheng, V.S.S.; Cui, Z.; Zhou, X.; Xiong, H. Recurrent Convolutional Neural Network for Sequential Recommendation. In Proceedings of the World Wide Web Conference 2019, San Francisco, CA, USA, 13–17 May 2019; pp. 3398–3404.
7. Chatzis, S.P.; Christodoulou, P.; Andreou, A.S. Recurrent latent variable networks for session-based recommendation. In Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems, Como, Italy, 27 August 2017; pp. 38–45.
8. Tang, J.; Wang, K. Personalized top-n sequential recommendation via convolutional sequence embedding. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, Marina Del Rey, CA, USA, 5–9 February 2018; pp. 565–573.
9. Yuan, F.; Karatzoglou, A.; Arapakis, I.; Jose, J.M.; He, X. A Simple Convolutional Generative Network for Next Item Recommendation. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, Melbourne, Australia, 11–15 February 2019; pp. 582–590.

10. Zheng, G.; Zhang, F.; Zheng, Z.; Xiang, Y.; Yuan, N.J.; Xie, X.; Li, Z. DRN: A deep reinforcement learning framework for news recommendation. In Proceedings of the 2018 World Wide Web Conference on Wolrd Wide Web, Lyon, France, 23–27 April 2018; pp. 167–176.
11. Kang, W.C.; McAuley, J. Self-attentive sequential recommendation. In Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), Singapore, 17–20 November 2018; pp. 197–206.
12. Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; Jiang, P. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 19–23 October 2019; pp. 1441–1450.
13. Wang, Y.; Wang, L.; Li, Y.; He, D.; Liu, T.Y. A Theoretical Analysis of NDCG Type Ranking Measures. In Proceedings of the 26th Annual Conference on Learning Theory, Princeton, NJ, USA, 12–14 June 2013; pp. 25–54.
14. Portugal, R.L.Q.; Casanova, M.A.; Li, T.; do Prado Leite, J.C.S. GH4RE: Repository Recommendation on GitHub for Requirements Elicitation Reuse. In Proceedings of the 29th International Conference on Advanced Information Systems Engineering (CAiSE), Essen, Germany, 12–16 June 2017; pp. 113–120.
15. Sun, X.; Xu, W.; Xia, X.; Chen, X.; Li, B. Personalized project recommendation on GitHub. *Sci. China Inf. Sci.* **2018**, *61*, 050106. [CrossRef]
16. Shao, H.; Sun, D.; Wu, J.; Zhang, Z.; Zhang, A.; Yao, S.; Liu, S.; Wang, T.; Zhang, C.; Abdelzaher, T. paper2repo: GitHub Repository Recommendation for Academic Papers. In Proceedings of The Web Conference 2020, Taipei, Taiwan, 20–24 April 2020; pp. 629–639.
17. Zhang, L.; Zou, Y.; Xie, B.; Zhu, Z. Recommending Relevant Projects via User Behaviour: An Exploratory Study on Github. In Proceedings of the 1st International Workshop on Crowd-Based Software Development Methods and Technologies, Hong Kong, China, 17 November 2014; pp. 25–30.
18. Matek, T.; Zebec, S.T. GitHub open source project recommendation system. *arXiv* **2016**, arXiv:1602.02594.
19. Xu, W.; Sun, X.; Hu, J.; Bin, L. REPERSP: Recommending Personalized Software Projects on GitHub. 2017. In Proceedings of the International Conference on Software Maintenance and Evolution (ICSME), Shanghai, China, 17–24 September 2017; pp. 648–652.
20. Xu, W.; Sun, X.; Xia, X.; Chen, X. Scalable Relevant Project Recommendation on GitHub. In Proceedings of the 9th Asia-Pacific Symposium on Internetware, Shanghai, China, 23 September 2017; pp. 1–10.
21. Zhang, Y.; Lo, D.; Kochhar, P.S.; Xia, X.; Li, Q.; Sun, J. Detecting similar repositories on GitHub. In Proceedings of the 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), Klagenfurt, Austria, 20–24 February 2017; pp. 13–23. [CrossRef]
22. Zhang, P.; Xiong, F.; Leung, H.; Song, W. FunkR-pDAE: Personalized Project Recommendation Using Deep Learning. *IEEE Trans. Emerg. Top. Comput.* **2018**, 1.10.1109/TETC.2018.2870734. [CrossRef]
23. Rajaraman, A.; Ullman, J.D. *Mining of Massive Datasets*; Cambridge University Press: Cambridge, MA, USA, 2011.
24. Defferrard, M.; Bresson, X.; Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 3844–3852.
25. Davidson, J.; Liebald, B.; Liu, J.; Nandy, P.; Van Vleet, T.; Gargi, U.; Gupta, S.; He, Y.; Lambert, M.; Livingston, B.; others. The YouTube video recommendation system. In Proceedings of the Fourth ACM Conference on Recommender Systems, Barcelona, Spain, 26–30 September 2010; pp. 293–296.
26. Koren, Y.; Bell, R. Advances in collaborative filtering. *Recomm. Syst. Handb.*. **2015**, 77–118. Available online: https://link.springer.com/chapter/10.1007/978-1-4899-7637-6_3 (accessed on 10 Febraury 2021).
27. Guo, Y.; Wang, M.; Li, X. An interactive personalized recommendation system using the hybrid algorithm model. *Symmetry* **2017**, *9*, 216. [CrossRef]
28. Salakhutdinov, R.; Mnih, A.; Hinton, G. Restricted Boltzmann machines for collaborative filtering. In Proceedings of the 24th International Conference on Machine Learning, Corvalis, OR, USA, 20–24 June 2007; pp. 791–798.
29. Sedhain, S.; Menon, A.K.; Sanner, S.; Xie, L. Autorec: Autoencoders meet collaborative filtering. In Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015; pp. 111–112.
30. Wu, Y.; DuBois, C.; Zheng, A.X.; Ester, M. Collaborative denoising auto-encoders for top-n recommender systems. In Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, 22–25 February 2016; pp. 153–162.
31. Rendle, S.; Freudenthaler, C.; Schmidt-Thieme, L. Factorizing personalized markov chains for next-basket recommendation. In Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–30 April 2010; pp. 811–820.
32. He, R.; McAuley, J. Fusing similarity models with markov chains for sparse sequential recommendation. In Proceedings of the IEEE 16th International Conference on Data Mining (ICDM), Barcelona, Spain, 12–15 December 2016; pp. 191–200.
33. Zhang, C.; Wang, K.; Yu, H.; Sun, J.; Lim, E.P. Latent factor transition for dynamic collaborative filtering. In Proceedings of the 2014 SIAM International Conference on Data Mining, Philadelphia, PA, USA, 24–26 April 2014; pp. 452–460.
34. Mikolov, T.; Karafiát, M.; Burget, L.; Černockỳ, J.; Khudanpur, S. Recurrent neural network based language model. In Proceedings of the Eleventh Annual Conference of the International Speech Communication Association, Makuhari, Japan, 26–30 September 2010; pp. 1045–1048.

35. Liu, Q.; Zeng, Y.; Mokhosi, R.; Zhang, H. STAMP: short-term attention/memory priority model for session-based recommendation. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 1831–1839.
36. Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; Ma, J. Neural attentive session-based recommendation. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore, 6–10 November 2017; pp. 1419–1428.
37. Kim, J.; Wi, J.; Jang, S.; Kim, Y. Sequential Recommendations on Board-Game Platforms. *Symmetry* **2020**, *12*, 210. [CrossRef]
38. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CO, USA, 4–9 December 2017; pp. 5998–6008.
39. Zhang, S.; Tay, Y.; Yao, L.; Sun, A. Next Item Recommendation with Self-Attention. *arXiv* **2018**, arXiv:1808.06414.
40. Kula, M. Spotlight. 2017. Available online: https://github.com/maciejkula/spotlight (accessed on 20 December 2020).
41. Koren, Y. Collaborative filtering with temporal dynamics. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June–1 July 2009; pp. 447–456.
42. Bharadhwaj, H.; Joshi, S. Explanations for temporal recommendations. *KI-Künstliche Intell.* **2018**, *32*, 267–272. [CrossRef]
43. Ying, H.; Zhuang, F.; Zhang, F.; Liu, Y.; Xu, G.; Xie, X.; Xiong, H.; Wu, J. Sequential recommender system based on hierarchical attention networks. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 13–19 July 2018; pp. 3926–3932.
44. Hidasi, B.; Karatzoglou, A. Recurrent neural networks with top-k gains for session-based recommendations. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018; pp. 843–852.
45. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian personalized ranking from implicit feedback. *arXiv* **2012**, arXiv:1205.2618.
46. Marung, U.; Theera-Umpon, N.; Auephanwiriyakul, S. Top-N recommender systems using genetic algorithm-based visual-clustering methods. *Symmetry* **2016**, *8*, 54. [CrossRef]
47. Pan, R.; Zhou, Y.; Cao, B.; Liu, N.N.; Lukose, R.; Scholz, M.; Yang, Q. One-class collaborative filtering. In Proceedings of the Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 502–511.
48. Tseng, P. Convergence of a block coordinate descent method for nondifferentiable minimization. *J. Optim. Theory Appl.* **2001**, *109*, 475–494. [CrossRef]
49. Christodoulou, P.; Chatzis, S.P.; Andreou, A.S. A variational latent variable model with recurrent temporal dependencies for session-based recommendation (VLaReT). In Proceedings of the 27th International Conference on Information Systems Development, Lund, Sweden, 22–24 August 2018; pp. 51–64.
50. Ferrari Dacrema, M.; Parroni, F.; Cremonesi, P.; Jannach, D. Critically Examining the Claimed Value of Convolutions over User-Item Embedding Maps for Recommender Systems. In Proceedings the 29th ACM International Conference on Information & Knowledge Management, online, 19–23 October 2020; pp. 355–363.
51. Yu, J.; Yin, H.; Li, J.; Wang, Q.; Hung, N.Q.V.; Zhang, X. Self-Supervised Multi-Channel Hypergraph Convolutional Network for Social Recommendation. *arXiv* **2021**, arXiv:2101.06448.