International Journal of Data Science and Analytics (2023) 16:217–235
https://doi.org/10.1007/s41060-022-00367-4

REGULAR PAPER

A probabilistic perspective on nearest neighbor for implicit recommendation

Domokos M. Kelen[1] · **Andras A. Benczúr**[1]

Received: 25 January 2022 / Accepted: 7 October 2022 / Published online: 29 October 2022

Abstract
Over the past years, the recommender systems community invented several novel approaches that reached better and better prediction accuracy. Sequential recommendation, such as music recommendation, has seen large improvements from neural network-based models such as recurrent neural networks or transformers. When no sequential information is available or not relevant, such as for book, movie, or product recommendation, however, the classic k-nearest neighbor algorithm appears to remain competitive, even when compared to much more sophisticated methods. In this paper, we attempt to explain the inner workings of the nearest neighbor using probabilistic tools, treating similarity as conditional probability and presenting a novel model for explaining and removing popularity bias. First, we provide a probabilistic formulation of similarity and the classic prediction formula. Second, by modeling user behavior as a combination of personal preference and global influence, we are able to explain the presence of popularity bias in the predictions. Finally, we utilize Bayesian inference to construct a theoretically grounded variant of the widely used inverse frequency scaling, which we use to mitigate the effect of popularity bias in the predictions. By replacing the formerly ad hoc choices of nearest neighbor with probabilistically founded counterparts, we are able to improve prediction accuracy over a variety of data sets and gain an increased understanding of the theory behind the method.

Keywords Recommender systems · Implicit feedback · Nearest neighbor · User modeling · Bayesian inference · Ranking

1 Introduction

When recommending books, movies, or certain products in subscription-based services, the long- or mid-term user profile plays a more important role than the sequence of the last few items consumed, which is typically available in non-subscription-based services. While short-term consumption products that depend more on the current mood of the user such as music, products that the users purchase less frequently and potentially engage with for a longer time [1] are characterized better by looking back at past user behavior. Although neural models have greatly improved performance for the short-term, sequential recommendation compared with traditional approaches, they overemphasize the sequential information of sessions, that is, the order of items in each browsing session [2,3]. In contrast to session-based recommendation that predicts the next click or item for an ongoing session without knowing who is now performing the actions, in this paper, we concentrate on recommendation based on the long-term user profile, a task of practical relevance in itself or in a recommender ensemble combined with session-based methods.

In addition to focusing on recommendation based on long-term user profiles, we address the implicit recommendation task where models rely on interaction feedback such as viewing an item or listening to a song, and only the lack of interaction serves as negative feedback. Several practitioners argue that most of the recommendation tasks they face are implicit feedback [4]; in [5], the authors claim that 99% of the recommendation industry tasks are implicit. Even if sufficiently many users are willing to explicitly rate the items such as Netflix movies [6], much more detailed information is available in the form of implicit feedback of who interacted with which items. Even for Netflix movie ratings, the

✉ Domokos M. Kelen
kdomokos@sztaki.hu

Andras A. Benczúr
benczur@sztaki.hu

[1] Institute for Computer Science and Control (SZTAKI), Eötvös Loránd Research Network (ELKH), Kende u. 13-17, Budapest 1111, Hungary

implicit feedback version "who rated what" is an interesting and practically important task [7]. Toward this end, we also consider explicit feedback research data by transforming ratings into interaction regardless of its value.

Our contribution to non-sequential implicit feedback recommendation is the mathematical understanding and improvement of nearest neighbor, one of the oldest and most popular approaches for collaborative filtering [8]. While a multitude of more sophisticated methods has been developed over the years [9–11], very few can match the simplicity of the classic k-nearest neighbor. This simplicity also makes the recommendations and behavior of the method easier to interpret than many more recent counterparts, which makes it attractive in practice. Recently, several thorough experiments proved that they are still often very competitive in terms of predictive performance [12], even when challenged by much more complex methods, and in some cases, even compared to sequential recommendation recurrent network methods [13].

Despite the success of the nearest neighbor recommender, there is a lack of theoretical analysis and understanding of the foundations of nearest neighbor methods for implicit feedback recommendation in the literature. Already in 2007, Bell and Koren [14] listed various concerns regarding certain theoretically unjustified components, including the lack of statistical foundations for the similarity functions and the difficulty to avoid overfitting for similarities with low-support items.

Our goal is to provide a probabilistic formulation, which has multiple uses. A probabilistic basis allows us to understand the behavior and properties of the algorithm in much more detail. Further, our work provides insight into not only the mechanics of the algorithm we investigate but of the data itself, both of which could potentially influence or inspire future research.

Even the very recent nearest neighbor implementations [15–17] depend on ad hoc choices and are heuristic in nature. In particular, the theoretical foundation is missing for the similarity functions [14]. Understanding machine learning approaches in probabilistic theory can become challenging, as the prediction tasks quickly become too complex for traditional statistical machinery to sufficiently handle. Oversimplification can lead to failing to account for certain effects, which may result in the approach under-performing its heuristic counterparts. Accounting for these effects often requires nontrivial methods. In our approach, we investigate frequently used heuristics and reinterpret them in a probabilistic framework.

Prior to our work, research mostly focused on learning the similarities [14,18] and there have been only a few attempts for a probabilistic interpretation of the nearest neighbor algorithm, mostly for the classification problem [19,20]. While probabilistic analysis of nearest neighbor for explicit ratings exists [21], we are aware of no probabilistic formulations in the implicit case on how interactions with neighbors affect the preference of the item in question, which can also be combined with separating the popularity component of the implicit feedback.

Our work reinterprets the item-based nearest neighbor recommendation algorithm by considering similarity a conditional probability. We frame multiple heuristics as ways to deal with the uncertainty of the observed similarity and propose using the Bayesian credible interval lower bound as an alternative. For the selected set of neighbors, we argue that the classic prediction formula is in fact a probabilistic OR operation over multiple predictors, using the assumption that these predictors are independent. Although the independence assumption is invalid in general [14], empirical tests still confirm in practice; additionally, methods to improve independence in the selected neighborhood remain an option for future work.

We also show that it is plausible that the independence assumption is responsible for introducing a strong popularity bias in the predictions. Our modeling of the popularity component in item similarity explains the popularity bias that has been observed many times either directly [22] or through the need for inverse frequency scaling [23–25].

Decoupling popularity from similarity is a key component of our work, which leads to more recommendations in the long tail without sacrificing accuracy. We give a Bayesian interpretation of the user behavior as a combination of popularity and personal taste. We consider each positive feedback observation as an OR operation over a popularity and a personal taste component. By assuming a Beta prior on the personal taste component, we derive a Bayesian formula for the posterior, which we then use in the nearest neighbor recommendation. Using this approach, we give a theoretically founded method to reduce the effect of selection bias toward the popular items when measuring personal taste.

Our model relies on a Bayesian interpretation of the user behavior as a combination of popularity and personal taste. We consider each positive feedback observation as an OR operation over a popularity and a personal taste component. By assuming a Beta prior on the personal taste component, we derive a Bayesian formula for the posterior, which we then use in the nearest neighbor recommendation. Using this approach, we give a theoretically founded method to reduce the effect of selection bias toward the popular items when measuring personal taste.

As a summary, our key contributions toward a probabilistic interpretation of the nearest neighbor method are as follows:

- We reinterpret similarity as a conditional probability.
- We introduce the Bayesian credible interval lower bound to incorporate uncertainties.

- We decouple popularity from personal taste by using Bayesian inference.

Limitations of our approach include the assumption that feedback is binary: We do not take into account the number of interactions between a given user-item pair, only the fact of whether it exists. Further, in this paper, we only consider the case of item-based nearest neighbor recommendations, however, most of the ideas presented could carry over to user-based nearest neighbor recommendation as well.

We experiment with a large variety of test data and compare to the recent baseline implementations of Dacrema et al. [12]. We find that our approach works well in practice, providing a notable recommendation accuracy increase over other item-based nearest neighbor approaches, while still closely matching the structure of the original method. Even though our goal is first and foremost to better understand the mechanics of the nearest neighbor method, this improvement also strongly supports the validity of our claims.

This paper is organized as follows. After the related results, we give a brief overview of the existing main nearest neighbor approaches. The probabilistic interpretation and our new nearest neighbor algorithm are found in Sect. 4, and in Sect. 5, we describe our evaluation methodology and the conducted experiments.

2 Related results

Recommender systems [26] have become extremely common recently in a variety of areas including movies, music, news, books, and products in general. They produce a list of recommended items by either collaborative or content-based filtering. Collaborative filtering methods [27,28] build models of the past user-item interactions, while content-based filtering [29] typically generates lists of similar items.

In the past few years, several papers addressed the so-called *sequential recommendation* problem where the recommendation is based on the last sequence of items in a user session. This task is highly relevant when the users are reluctant to create logins and prefer to browse anonymously. Most popular and successful algorithms use recurrent neural networks such as [30–35], however, there are other approaches as well, such as using graph neural networks [36]. A large-scale empirical evaluation of session-based recommendation algorithms [13] enumerates and compares these algorithms. Note that, the main conclusion of that study is that "the progress in terms of prediction accuracy that is achieved with neural methods is still limited. In most cases, our experiments show that simple heuristic methods based on nearest neighbors schemes are preferable over conceptually and computationally more complex methods" [13].

Even more recently, several authors noticed that sequential recommendation provides suboptimal results as they cannot sufficiently capture long-term user preferences [2,3,37,38]. While most of these papers concentrate on improving sequence recommenders by engineering features of past user interactions into their models, we believe that addressing the non-sequential component alone is an important contribution, since the best methods can then be combined in an ensemble recommender [39].

In this paper, we consider user independent **item-to-item** recommendation based on the profiles of the entire past user interactions [27,28]. The best known example of this task is the Amazon list of books related to the last visited one [27]. We model the implicit feedback of the user. The feedback may be explicit, such as one to five stars ratings of movies on Netflix [40]. Most of the recommendation tasks are however **implicit**, as the user provides no like or dislike information. In such cases, recommender systems have to rely on implicit feedback such as time elapsed viewing an item or listening to a song. Although implicit recommendation is much more frequent in practice, explicit feedback is overrepresented in research data. For this reason, in our experiments, we consider a few explicit recommendation research data by transforming a rating into an interaction regardless of its value.

Even in the case when explicit feedback is available, it is only for a small fraction of the interactions, and due to the frugality of the explicit feedback, it is important to know who rated what [7]. Furthermore, each user often takes some implicit actions (e.g., click) before making an explicit decision (e.g., purchase), hence the binarized, implicit feedback version of the original explicit feedback data is of high relevance [41]. Note that, BPR [42] also turns both explicit and implicit feedback into user-dependent pairwise preferences regarding items. In our experiments, we consider a binarized version of the explicit feedback data of Amazon, MovieLens, Epinions, and more.

The first item-to-item recommender methods [27,28,43] were using similarity information to directly find nearest neighbor transactions. Very early papers [44] already compared the performance of several variants; for a more recent one, see [45]. The two main variants are based either on user [43] or item [28] similarities; in [15], a reformulation is proposed that unifies the two approaches. Nearest neighbor algorithms differ in the similarity functions used [18,46], in feature weighting [23], and other normalization approaches [14]; some apply spectral methods to denoise the similarity matrix [17] or use different framings of item-to-item approaches [47].

However, the theory behind nearest neighbor methods was criticized for at least two main reasons. First, the similarity metrics typically have no mathematical justification. Second, the confidence of the similarity values is often not involved

when finding the nearest neighbor, which leads to overfitting in sparse cases. In [14], a method is given that learns similarity weights for users. Their method estimates user preference toward an item by regression over similar item ratings. By the nature of the regression, their method applies to explicit feedback recommendation only. One of our main goals is to provide an implicit feedback counterpart to their result, albeit based on a probabilistic approach for implicit feedback recommendation.

Many works introduce probabilistic reasoning for various collaborative filtering approaches that use elements similar to ours. The notion of using conditional probability as similarity for nearest neighbor appears as early as [22]. In that work, the author already notes that the approach is known to lead to popularity bias and mentions TF-IDF and normalization as approaches to mitigate; however, the proposed solution is based on a heuristic power function transformation of the conditional probability inspired by TF-IDF rather than a theoretically founded method. In [48], an undirected graphical model over the items is introduced with a fast training algorithm to learn the relations between items; however, there the main goal is to extend the heuristic solutions by neighbor interaction considerations. The work [49] computes probabilities for how likely two users are to be of the same personality type. Their method can be considered as a Naive Bayes network with past positive user interactions as features; however, their method makes no consideration to handle the difference between sparse and dense observations. The paper [50] also deals with user behavior and collaborative filtering in a Bayesian setting; however, their work ventures rather far from the classic KNN formulation.

Of special note is the recent work [21] that has a very similar theme to ours, attempting to place KNN-based recommendation in a probabilistic framework, and also using a conditional probability-based explanation of similarity. However, the basic approach to problem formulation is different, with [21] marginalizing over categorical variables rather than treating events as binomial variables. As a result, the two approaches further diverge in how they treat ensemble prediction and related assumptions. We believe our formulation is more straightforward, while also allowing for a much more direct way of modeling popularity bias, and an intuitive solution to handling it through Bayesian inference. Our model also has the additional benefit of providing insight into the underlying user behavior.

One of our main theoretical and practical questions relates to separating popularity from personal taste in the feedback. Most authors address this question in the context of the missing-not-at-random feedback [51–55], however, they all take approaches different from ours. For example, in [56], the training procedure itself is de-biased. In [54], a thorough analysis is given on item popularity in recommendation, which focuses primarily on evaluation aspects such as the fraction of recommendation in the long tail. In [57], a probabilistic latent variable model is proposed to make use of the sparse signals from the users. Finally, for explicit ratings, the behavior on missing feedback is probabilistically modeled in [55], however, it remains unclear how their model can be applied for implicit feedback. Since in our research, we have neither ground truth nor testing user base, we restrict our attention to the user behavior as observed in the data. Popularity bias in the data does not primarily affect the traditional offline recommendation accuracy metrics such as recall or NDCG, as the bias is present in the test set just as much as in the training set. Rather, it hinders recommendation in the long tail, which is considered particularly valuable for the users [58]. In our work, we are concerned with the bias introduced by the method as also observed in [22], instead of the bias present in the collected data.

Regarding popularity bias, as we discussed, several authors, for example [56,57], attempt to de-bias the data itself to directly provide a better user experience. While several heuristics (regularization and shrink [14], popularity-stratified training [54], confidence weight [57], de-biasing the evaluation [56], etc.) were proposed to account for popularity, we require a well-grounded formulation for combining popularity and personal taste.

3 Background

In this section, we give an overview of the main variants of the classic item-based KNN using cosine similarity [28, 59]. We mention that the first variants were user similarity-based [43] and user-based approaches also perform well in recent experiments [12]. While our ideas could carry over to the user-based nearest neighbor problem, we restrict to item similarity for the clarity of the presentation.

Nearest neighbor algorithms typically make an ad hoc choice of a similarity measure, which is only empirically justified. For example, different papers propose the Jaccard coefficient [18], Cosine [28], Asymmetric Cosine [46], and others such as Dice-Sorensen and Tversky similarities [12]. In some variants, we can consider the denominator in the similarity measure as a normalization term. Next, we describe Cosine similarity as an illustration and motivation for our definition of similarity as a conditional probability.

The item-based KNN algorithm, which is the main focus of this paper, uses a number of heuristics. First, the user-item interaction matrix may optionally be transformed using TF-IDF (or BM25) normalization [23]:

$$z_{u,i}^{\text{tfidf}} = \frac{\sqrt{z_{u,i}}}{\log\left(\frac{M}{t_i}\right)}. \tag{1}$$

Here, $z_{u,i} \in \{0, 1\}$ is the observation of whether user u interacted with item i, M is the number of items, and $t_i = \sum_u z_{u,i}$. Since we assume the matrix to be $0-1$ valued, in TF-IDF, the square root in the numerator holds no significance.

Next, the similarity values are computed between items. The cosine similarity value for items i and j is calculated as

$$\cos(i, j) = \frac{\sum_u z_{u,i} z_{u,j}}{\sqrt{t_i}\sqrt{t_j}}. \tag{2}$$

Optionally, this calculation involves a shrink value [14] to penalize items with too few observed interactions:

$$\text{sim}(i, j) = \frac{\sum_u z_{u,i} z_{u,j}}{\sqrt{t_i}\sqrt{t_j} + \text{shrink}}. \tag{3}$$

Next, for each item i, the k items $n(i) = \{n(i)_1, \ldots, n(i)_k\}$ with the highest similarity values are selected.

Finally, we calculate our prediction $\hat{z}_{u,i}$ for each item i and rank the items to produce a toplist of the items that the user is most likely to interact with by

$$\hat{z}_{u,i} = \sum_{j \in n(i)} \text{sim}(i, j)\, z_{u,j}. \tag{4}$$

Note that, we use certain similarity values for ranking neighbors and other values in the prediction formula of Eq. (4). The similarity values in the two cases do not necessarily have to be the same, but they are often selected so. In our work, however, we give separate interpretations for the two subproblems and use different similarity values.

4 Our new algorithm in the probabilistic interpretation of nearest neighbors

In this section, we introduce our main result, a probabilistic interpretation of the KNN method. The key ingredients are summarized as follows:

- We interpret similarity as a conditional probability and propose using Bayesian credible intervals to select neighbors.
- We present a plausible explanation for the neighbor ensemble prediction formula of the classic KNN method.
- We explain the presence of popularity bias in the predictions and develop a formula to mitigate its effects by using Bayesian inference to acquire an estimate of popularity-free personal taste. Our Bayesian approach is reminiscent of Latent Dirichlet Allocation, however, with a simpler distribution (binomial instead of multinomial) but with a more complex inference that involves an operation between the random variables expressing item popularity and user personal taste.

The notation used throughout this section is summarized in Table 1.

We summarize the outline of this Section along with our main contributions as follows:

- In Sect. 4.1, we replace the naive formulation of the observed conditional probability of a new interaction given a past one by a Bayesian inference using Beta priors.
- In the same section, we introduce Bayesian credible intervals to handle the uncertainties if too few interactions are available in the data.
- In Sect. 4.2, using mild independence assumptions, we derive formula (11) for computing the OR of all past interactions that indicate a given new one and show how this formula can efficiently be computed.
- In Sect. 4.3, we also involve negative events (items not consumed by the user) in our formulation.
- In Sect. 4.4, we enhance the main formula (11) to decouple popularity from user preference. We introduce and handle separate taste and popularity components of a given user interaction.
- The decoupled components of the interaction are handled separately in the next two subsections. In Sect. 4.5, we introduce a variant of the estimated or maximum a posteriori estimates to handle the taste component and mathematically prove its properties.
- In Sect. 4.6, we complete our formula by adding the popularity component as a Bernoulli variable.

4.1 Conditional probability as similarity

In an attempt to probabilistically interpret the influence of the nearest neighbor items j on the unknown feedback of an item i, let us interpret each observation $z_{u,i}$ as a single sample taken from a Bernoulli variable $Z_{u,i}$. Our goal is then to predict $P(Z_{u,i} = 1)$ for each i to produce a ranking list. Typical methods select a set $n(i)$ of items j, for example those with strongest similarity to i, as an "expert committee" to decide on i.

To select neighbors $n(i)$ with the highest predictive power, we want to rank them by the conditional probability $P(Z_{u,i} = 1 \mid Z_{u,j} = 1)$. A naive way to estimate this value from the data is to calculate

$$P(Z_{u,i} = 1 \mid Z_{u,j} = 1) \approx \frac{\sum_u z_{u,i} z_{u,j}}{t_j}. \tag{5}$$

However, this formula fails to account for the uncertainty of the estimation, i.e., the sample size t_j. Rather, we propose estimating this probability as the parameter of a Beta-binomial variable with sample size t_j and a number of observed positive outcomes equal to $\sum_u z_{u,i} z_{u,j}$. Since the

Table 1 Summary of notations

u	User
i, j	Items
t_i	Positive feedback count on item i
$z_{u,i}$	Observed interactions between users and items
$Z_{u,i}$	The Bernoulli variables that we assume $z_{u,i}$ are sampled from
$X_{u,i}$, $Y_{u,i}$	The user specific and global Bernoulli variable components of $Z_{u,i}$, respectively
$x_{u,i}$, $y_{u,i}$	Parameters of $X_{u,i}$ and $Y_{u,i}$
$n(i)$	Set of k selected neighbors for item i
$n_u^+(i)$	Set of k selected neighbors for item i for which $z_{u,j} = 1$
$Z_{u,j} \twoheadrightarrow Z_{u,j}$	Assumed causal events where consuming j signals an independent reason for u to consume i
$\|U\|$, $\|I\|$	Number of users and items
q	Confidence value for the credible interval in Eq. (8)
c	Popularity scaling coefficient in Eq. (22)
α	Weight of popularity after recombining with the popularity-free decoupled estimate in Eq. (41)

Note the difference in the relation of x, y to X, Y (parameters) versus z to Z (observation)

observable events are binary, the Beta distribution is a natural and often used choice of prior for the parameter of the resulting binomial variables, as it has the convenient property of being a conjugate prior, i.e., the posterior also assumes a Beta distribution. Our estimate thus becomes

$$P(Z_{u,i} = 1 \mid Z_{u,j} = 1) \approx \frac{\sum_u z_{u,i} z_{u,j} + a}{t_j + b}, \tag{6}$$

where a and b are the parameters of the Beta-prior used. Note that, if our prior assumes a low probability, i.e., a is significantly smaller than b, then this closely resembles the shrink heuristic mentioned in Sect. 3.

To emphasize the importance of sample size, we can calculate a Bayesian credible interval for the posterior distribution and rank the items based on the lower interval bound. This effectively means that we discount items by the uncertainty present in our estimate, i.e., we calculate the lower bound $\gamma_{u,i}$ for which

$$P(\ P(Z_{u,i} = 1 \mid Z_{u,j} = 1) < \gamma_{u,i}) = q, \tag{7}$$

where q is a given confidence value, for example $q = 0.05$. With the assumed Beta prior, the value of q is given by

$$\gamma_{u,i} = B^{-1}\left(\ q\ ,\ t_{i,j} + a\ ,\ t_j - t_{i,j} + b\right), \tag{8}$$

where $t_{i,j} = \sum_u z_{u,i} z_{u,j}$ and $B^{-1}(\cdot, a, b)$ denotes the inverse of the cumulative distribution function of a Beta distribution with parameters a and b. To compute, we use the numerical approximation implemented by the Python package *SciPy* [60] and a lookup table for avoiding multiple calculations of the same value.

4.2 Ensemble prediction

Next, we propose a probabilistic interpretation for aggregating the contribution of similar items in predicting user taste. In the literature [14], several authors argue that there is no theoretical foundation for the similarity measure and the summation of Eq. (4).

Our reasoning starts with defining a not directly observable event that the reason for user u consuming item i is an earlier item j, or at least j is indicative of some underlying reason, such as a certain component of the user's personal taste. In other words, if $Z_{u,j} = 1$, we may in this case infer $Z_{u,i} = 1$ with certain probability. We denote this abstract event as $Z_{u,j} \twoheadrightarrow Z_{u,i}$. In this formulation, the user will consume item i if there is any similar item $j \in n_u^+(i)$ for which $Z_{u,j} \twoheadrightarrow Z_{u,i}$ is true, i.e., if the following formula is true:

$$\bigvee_{j \in n_u^+(i)} Z_{u,j} \twoheadrightarrow Z_{u,i}, \tag{9}$$

where $n_u^+(i)$ is the set of neighbors of i that appear in the interaction history of user u.

Next, we approximate the probability that at least one neighboring item serves as a reason for u to consume i. Assuming that the events $Z_{u,j} \twoheadrightarrow Z_{u,i}$ are fully independent for $j \in n_u^+(i)$ and approximating

$$P(Z_{u,j} \twoheadrightarrow Z_{u,i}) \approx P(Z_{u,i} = 1 \mid Z_{u,j} = 1), \tag{10}$$

this yields

$$P\left[\bigvee_{j \in n_u^+(i)} Z_{u,j} \twoheadrightarrow Z_{u,i}\right]$$

$$\approx 1 - \prod_{j \in n_u^+(i)} 1 - P(Z_{u,i} \mid Z_{u,j} = 1). \tag{11}$$

Note that, by expanding the product on the right of Eq. (11), we get a formula that can be re-written to another form, also resulting from the inclusion-exclusion principle, such that the summation of Eq. (4) clearly appears as the first term:

$$\begin{aligned} & 1 - \prod_{j \in n_u^+(i)} (1 - P(Z_{u,i} \mid Z_{u,j} = 1)) \\ & = 1 - \left(1 - \sum_{j \in n_u^+(i)} P(Z_{u,i} \mid Z_{u,j} = 1) + \Delta \right) \\ & = \sum_{j \in n(i)} \text{sim}(i, j)\, z_{u,j} - \Delta. \end{aligned} \tag{12}$$

The rest of the terms Δ prove to be negligible in practice, as we show in Sect. 5.4. Similar to Sect. 4.1, we treat the conditional probability as the parameter of a Beta-binomial variable and estimate it using the formula (6).

It is important to note that the independence assumption is overly strong and most likely does not hold in realistic scenarios. At the same time, the predictive performance of the nearest neighbor methods indicates that this approximation works in practice. A consequence of this line of thinking is that one should try to select neighbors that are not only strong predictors but also independent of each other, as also observed in [14]. Equation (11) and the independence assumption will be further discussed in Sect. 4.4.

Another implicit assumption behind Eq. (11) as a prediction for $P(Z_{u,i})$ is that we have found and taken into account *all* possible reasons for the user to consume i, which is unreasonable in general. However, it makes sense to view Eq. (11) as the probability that the interaction happened *because* one of the modeled events caused it to happen. In essence, we need to view the value defined in Eq. (11) with the caveat that it is the probability that can be reasonably inferred based on the evidence taken into account (i.e., the top k neighbors) and ignoring any other event not explicitly modeled. Note that, in Sect. 4.4, we expand the modeled events to include a general item-dependant one. Further, this problem is alleviated by the fact that it is equally true for all items that we are trying to rank, thus the limited amount of evidence is not a problem when we compare candidates for recommendation.

Although it may not be immediately clear, formula (11) can be implemented with the same computational complexity as (4). The predictions in Eq. (4) are usually calculated by first computing an item-item neighborhood matrix A, where

$$\forall i \, \forall j \in n(i) : A_{j,i} = sim(i, j) \tag{13}$$

and $A = 0$ elsewhere. The calculation of Eq. (4) then becomes a vector-matrix multiplication between the $0and1$ valued user interaction vector and the sparse matrix A. By transforming Eq. (11) into an equivalent formula

$$1 - \exp\left(\sum_{j \in n(i)} z_{u,j} \, \log\left(1 - P(Z_{u,j} \twoheadrightarrow Z_{u,i})\right) \right), \tag{14}$$

we can calculate predictions as a matrix-vector multiplication.

4.3 Predicting from negative events

For the ensemble formula in Eq. (9), we used predictors in the form

$$Z_{u,j} \twoheadrightarrow Z_{u,i}. \tag{15}$$

By the above equation, our method for selecting the nearest neighbors essentially means that we select events that we can use to predict the event $Z_{u,i} = 1$. However, so far, we only considered events in the form $Z_{u,j} = 1$. It is also possible to consider events in the form $Z_{u,j} = 0$, meaning that we also incorporate users *not* interacting with an item to predict new interactions.

To assess the power of predictors based on negative events, we can use a formula similar to the one used in Eq. (6), i.e.,

$$\begin{aligned} & P(Z_{u,i} = 1 \mid Z_{u,j} = 0) \\ & \approx \frac{\sum_u z_{u,i}(1 - z_{u,j}) + a}{|I| - t_j + b}. \end{aligned} \tag{16}$$

We measure the estimated strength of negative predictors and their effect in Sect. 5. We note in advance that using negative predictors is not very promising, as the basic premise of the mechanics described in Sect. 4.2 is that the predictor event has a causal relationship with the predicted event. Such a relationship is much harder to assume when the predictor event is a missing interaction.

4.4 Inverse frequency scaling

Next, we infer the popularity component of taste by introducing a new unobservable event $Y_{u,i}$ if a user u interacts with item i due to its popularity. We apply Bayesian inference for modeling the relation of popularity and personal taste by assuming that the user interacts with the item either because of personal motivation or because of popularity.

The intuition behind the usage of TF-IDF in Eq. (1) is that interacting with a very popular item carries less information about the user's taste than interacting with a less frequent item. We model this intuition by assuming that observations

$z_{u,i}$ arise as the combination of multiple independent components of behavior

$$Z_{u,i} = X_{u,i} \vee Y_{u,i}, \tag{17}$$

where $X_{u,i}$ is the Bernoulli variable of u interacting with i because of personal motivation, $Y_{u,i}$ is the Bernoulli variable of any particular user interacting with i because of its popularity, and $Z_{u,i}$ is the variable that we are able to observe in the form of $z_{u,i}$ samples. We assume X and Y to be independent by definition, as we view Y as common behavior and X as deviation from common behavior. Further, we assume $Y_{u,i}$ to be independent of u and denote it by Y_i (with its parameter denoted by y_i) in the remainder of the paper. Hence, the probability $P(Z_{u,i} = 1)$ given the values of $x_{u,i}$ and y_i can be expressed as

$$P(Z_{u,i} = 1 \mid x_{u,i}, y_i) = x_{u,i} + y_i - x_{u,i} \cdot y_i. \tag{18}$$

This ties back into the independence assumption of Eq. (11): If the observed variables $Z_{u,i}$ contain a component Y_i that is independent of all other $Z_{\cdot,j}$, then what we are actually estimating in Eq. (11) is instead

$$\begin{aligned} P(Z_{u,i} \mid z_{u,j} = 1) &\approx P(Z_{u,j} \twoheadrightarrow (X_{u,i} \vee Y_i)) \\ &= P((Z_{u,j} \twoheadrightarrow X_{u,i}) \vee Y_i). \end{aligned} \tag{19}$$

Thus, when in Eq. (11) we use the independence assumption to calculate

$$P(Z_{u,i}) \approx P\left[\bigvee_{j \in n_u^+(i)} Y_i \vee (Z_{u,j} \twoheadrightarrow X_{u,i})\right], \tag{20}$$

we introduce significant popularity bias, as we count the Y_i component multiple times. By this argument, if we increase the neighborhood size, we also increase popularity bias, which we will indeed measure in Sect. 5.6. To mitigate the effect of popularity in combination with neighborhood size, we propose using the formula

$$\begin{aligned} P(X_{u,i}) &\approx P\left[\bigvee_{j \in n_u^+(i)} Z_{u,j} \twoheadrightarrow X_{u,i}\right] \\ &\approx 1 - \prod_{j \in n_u^+(i)} (1 - P(X_{u,i} \mid Z_{u,j} = 1)) \end{aligned} \tag{21}$$

to get an estimate for the personal component $X_{u,i}$. We discuss estimating $P(X_{u,i} \mid Z_{u,j} = 1)$ in Sect. 4.5 and the reintroduction of Y_i into the prediction in Sect. 4.6.

4.5 Estimating $P(X_{u,i} \mid Z_{u,j} = 1)$

Estimating $P(X_{u,i} \mid Z_{u,j} = 1)$ is possible by treating $X_{u,i}$ as a Beta-binomial variable and doing inference for it based on Eq. (18). For this, we also need the value of y_i, which we estimate as

$$P(Y_i = 1) = \frac{ct_i}{|U|}; \tag{22}$$

in other words, we assume it to be proportional to the overall frequency of the item in the data divided by the number of users $|U|$, with a global scaling coefficient c to be able to adjust the strength of the assumed bias. We treat c as a hyperparameter of the method.

Statement 1 *Let X follow a binomial distribution with parameter x. Let us assume the relation of Eq. (17) between X, Y, Z. Then, assuming a Beta(a, b) prior distribution for X, we have*

$$f_{x|z}(x) = \frac{((1-y)x + y)^r \; x^{a-1} \; (1-x)^{b+s-1}}{\int_0^1 ((1-y)x_0 + y)^r \; x_0^{a-1} \; (1-x_0)^{b+s-1} dx_0}, \tag{23}$$

where r and s are the number of positive and negative samples observed, respectively.

Proof Since observing $z_{u,i}$ carries only indirect information about $X_{u,i}$, our posterior probability will no longer be Beta. Further, we need to calculate the posterior while observing all of the samples at the same time, otherwise the ordering of the samples would influence the resulting posterior. Thus, we need to calculate

$$\begin{aligned} f_{x|z}(x_0) &= \frac{P(z \mid x = x_0) f_x(x_0)}{P(z)} \\ &= \frac{P(z \mid x = x_0) f_x(x_0)}{\int_0^1 P(z \mid x = x_1) f_x(x_1) dx_1}, \end{aligned} \tag{24}$$

where z is the observed sample of $P(Z_{u,j} \mid X_{u,i})$ values and

$$f_x(x_0) = \frac{x_0^{a-1}(1-x_0)^{b-1}}{B(a,b)}. \tag{25}$$

Further,

$$P(z \mid x = x_0) = ((1-y)x_0 + y)^r ((1-x_0)(1-y))^s, \tag{26}$$

where r and s are the number of positive and negative samples observed. □

The number of positive and negative samples in our case is given by

$$r = \sum_u z_{u,i} z_{u,j} \tag{27}$$

$$s = t_j. \tag{28}$$

After deriving the density function for $P(X_{u,i} \mid Z_{u,j})$ in Statement 1, we turn to calculating the or relation of all past items that can potentially be the reason for user u consuming item i in Eq. (21). Since this equation for the or relation involves a product of probabilities, it is prohibitively complex to compute the distribution of $P(X_{u,i})$ by this equation. Instead, we approximate each $P(X_{u,i} \mid Z_{u,j})$ value by a point estimate first, such as the *estimated a posteriori* (EAP) or *maximum a posteriori* (MAP) estimate. Unlike in the case of a Beta distribution, in our case, these two estimates do not coincide and are nontrivial to calculate.

While an analytical formula of the EAP estimate exists for integers a and b, it is not practical to compute. Approximate integration can be done using various methods such as *self-normalized importance sampling*.

In our case, this means approximating the infinite sum

$$E(X_{ui} \mid Z_{uj}) \approx \frac{\sum_{\tilde{x}_i} \tilde{x}_i \frac{f_{x|z}(\tilde{x}_i)}{q(\tilde{x}_i)}}{\sum_{\tilde{x}_i} \frac{f_{x|z}(\tilde{x}_i)}{q(\tilde{x}_i)}} \text{ where } \tilde{x}_i \sim q, \tag{29}$$

by including only a finite number of terms. Note that, the denominator of $f_{x|z}$ and optionally of q can be canceled out for the computation, thus the integral itself does not need to be calculated.

The distribution q is the so-called *proposal distribution*, which can be any density function that is reasonably similar to the one we are trying to approximate. Since effective sampling methods for Beta distributed variables exist, we can choose q to be a Beta distributed with the shape parameters a_q and b_q chosen such that the maximum of the of the density function coincides with that of $f_{x|z}$. For this, we need the MAP estimate of $f_{x|z}$. Since in case of the beta distribution the expected value coincides with the maximum of the density function, we need to set

$$\frac{a_q}{a_q + b_q} = \text{MAP}(f_{x|z}). \tag{30}$$

The sample size parameter (i.e., $a_q + b_q$) is chosen as $p \cdot (a + b + r)$ where $p \in (0, 1]$ to make the computation more stable.

While the formula is guaranteed to converge, in our experience, a large number of samples (in the order of 10^4) are needed for stability. Since we need to calculate the estimate k times for each item i, the approximate integration proved to be too computationally expensive for our purposes. Because of this, we resort to using the MAP estimate itself

$$\arg\max_{x_0} f_{x|z} = \arg\max_{x_0} P(z \mid x = x_0) f_x(x_0), \tag{31}$$

which is relatively straightforward to compute by the next statement.

Statement 2 *Assuming $a > 2$ and $b + s > 2$, the MAP estimate for the distribution described in Eq. (23) can be written as*

$$\frac{-c_2 - \sqrt{c_2^2 - 4c_1c_3}}{2c_1}, \text{ where} \tag{32}$$

$$c_1 = (a' + b' + r)(y - 1) \tag{33}$$

$$c_2 = (a' + r)(y - 1) + (a' + b')y \tag{34}$$

$$c_3 = a'y \tag{35}$$

and using substitutions $a' = a - 1$ and $b' = b + s - 1$.

Proof We can find the maximum by taking the derivative of the nominator and finding its root in the (0, 1) interval. Note that, this root of the derivative is guaranteed to exist since the function is continuous, zero at both 0 and 1 and nonzero in between, thus it must have an extreme point in the interval (0, 1). Calculating the derivative is straightforward:

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}x_0}((1-y)x_0 + y)^r \; x_0^{a'} \; (1 - x_0)^{b'} \\ &= ((1-y)x_0 + y)^{r-1} \; x_0^{a'-1} \; (1 - x_0)^{b'-1} \omega(x_0), \end{aligned} \tag{36}$$

meaning roots at $-\frac{y}{1-y}$, 0, 1, and at the roots of the polynomial

$$\omega(x_0) = c_1 x_0^2 + c_2 x_0 + c_3. \tag{37}$$

We can observe that c_1 is always negative, as $a', b', r > 0$ and $0 < y < 1$, making the denominator of Eq. (32) also negative. Further, c_3 is clearly always positive. This in turn makes the expression

$$-c_2 + \sqrt{c_2^2 - 4c_1c_3} \tag{38}$$

positive, resulting in the corresponding root being negative, thus leaving the only possible candidate the one described in Eq. (32). □

We can also use a very similar calculation for estimating $P(X_{u,i} \mid Z_{u,j} = 0)$. In fact, Statements 1 and 2 still hold, we only need to use a different definition for the positive and negative samples r and s:

$$r = \sum_u z_{u,i}(1 - z_{u,j}), \tag{39}$$

$$s = |I| - t_j. \tag{40}$$

4.6 Reintroducing popularity Y_i

Finally, to get a prediction for $P(Z_{u,i} = 1)$, we have to reintroduce the effect of selecting item i due to its popularity, an abstract event that we denoted by $Y_{u,i}$.

In our next calculations, we assume that our inference for X might not be perfect in the sense that the prediction we calculate by Eq. (11) might still depend on Y_i. We model the remaining effect of Y_i on the inferred X_i using a Bernoulli variable A with parameter α:

$$P(X_{u,i} \vee (A \wedge Y_i)) = x_{u,i} + \alpha y_i - \alpha x_{u,i} y_i, \tag{41}$$

which can be transformed to the formula of our final algorithm using hyperparameter α:

$$\begin{aligned} P(Z_{u,i} = 1) &= x_{u,i} + y_i - x_{u,i} y_i \\ &= \left(x_{u,i} + \alpha y_i - \alpha x_{u,i} y_i\right) \cdot \frac{1 - y_i}{1 - \alpha y_i} \\ &\quad + \frac{y_i - \alpha y_i}{1 - \alpha y_i}. \end{aligned} \tag{42}$$

Optimal values for α are highly dependent on the values of other hyperparameters: Both the number of neighbors k and the global frequency scaling coefficient c in Eq. (22) heavily influence the amount of popularity bias in the recommendations.

4.7 Summary

To summarize the resulting method, we take the following steps when calculating a prediction for $P(Z_{u,i})$.

1. We calculate the credible interval $\gamma_{u,i}$ with the selected value α for each possible neighbor candidate using Equation (8).
2. We select the k top ranked neighbors as predictors.
3. We calculate the MAP estimate for $P(Z_{u,j} \twoheadrightarrow X_{u,i})$ for each selected neighbor using Statement 2.
4. By the MAP estimate, we return to Sect. 4.4 to calculate $P(X_{u,i})$ using the ensemble formula (21).
5. We obtain our final prediction score $P(Z_{u,i})$ by reintroducing the global popularity component using Equation (42).

Throughout this derivation, we rely on the following hyperparameters: the percentile q in step (1); the (possibly distinct) parameters for the Beta priors in steps (1) and (3); the global scaling parameter c in step (3); and α in step (5). We discuss hyperparameter selection in Sect. 5.9.

5 Experiments

5.1 Datasets and evaluation

We conduct our experiments on five explicit ratings datasets, the first three of which are also the ones used in [61], by the same transformation into an implicit recommendation task as in our experiment.

The datasets used are as follows:

- **Amazon Instant Video** [62]: ratings from Amazon.com. The records have been transformed by transforming a rating into an interaction regardless of its value. Users with fewer than 5 ratings were removed. The training-test and then the training-validation sets are split randomly on a per-user basis with given percentages (20% and 20%).
- **MovieLens 1M** [63]: a widely used dataset of movie ratings. Implicit transformation was done the same way as above. The training-test and then the training-validation sets are split randomly on a per-user basis with given percentages (20% and 10%).
- **HetRec** [64]: a dataset released by the Second International Workshop on Information Heterogeneity and Fusion in Recommender Systems. Implicit transformation was done the same way as above. The training-test and then the training-validation sets are split randomly on a per-user basis with given percentages (20% and 20%).
- **Epinions** [65]: This dataset is collected from Epinions.com website, which provides an online service for users to share product feedback. Implicit transformation was done the same way as above. The validation and test sets were each created by leaving one interaction out of the training set randomly for each user.
- **Amazon Books** [66]: book ratings from Amazon.com spanning May 1996–July 2014. To create an implicit dataset, we filtered for ratings of 5 and extracted the 10-core [67] of the remaining graph, resulting in a dataset that is still one of the most sparse of the ones used. The training-test-validation sets are created by randomly assigning every record with the given probabilities to each set (0.9, 0.05 and 0.05).

The datasets used have varying sizes and densities, see Table 2 for specifics. For measuring predictive performance, results are reported in terms of *Normalized Discounted Cumulative Gain* with a cutoff size of 50 (*N@50* or *NDCG@50*) and *recall* again with a cutoff size of 50 (*R@50* or *Recall@50*). We evaluate both metrics without sampling on item ranks that are calculated by considering the full item

Table 2 Statistics of the experimental data

	Users	Items	Interact.	Density	Valid./test size
A. Books	56257	50154	1418076	0.00050	0.05%/0.05%
A. Video	3113	5860	22184	0.00122	0.17%/0.19%
Hetrec	2044	5351	71680	0.00336	0.16%/0.20%
ML 1M	6013	3231	225473	0.00962	0.08%/0.20%
Epinions	40163	139738	664823	0.00012	0.05%/0.06%

set. We note that for this reason, the issues related to sampled ranking metrics described in [68] do not apply in our experiments.

These two metrics give a good indication of the performance of the methods, with NDCG focusing more on the top of the recommended list of items, and recall placing uniform weight on all positions.

5.2 Baselines

We use the baseline implementations and automatic hyperparameter tuning framework of [12,69] for measuring the performance of baseline algorithms. We optimize for NDCG@50 performance on the validation set in every case. Note that, the main purpose of including the best algorithms for a wide selection of model types is important to put our contribution in context, our primary objective is to improve upon the *Item KNN* algorithm as the key competitor. We also note that we only consider algorithms that use no item sequence information. For such tasks, our new algorithm can be evaluated in an ensemble combined with recurrent neural networks, as for example in [37] in future work.

We use the splitting procedure implemented by [12,69] for the *HetRec*, *Amazon Instant Video*, *Epinions* and *MovieLens 1M* datasets. The procedure is non-deterministic, thus the performance of different algorithms has some variance depending on the exact split generated. For this reason, we re-run the baselines on the same exact split that our models are also tested on. While not an exact match, the results we get are very much in line with the numbers reported in [12,69].

We also selected one dataset, the Amazon Books, which is not part of the evaluation framework of [12,69]. We split the interactions in this dataset randomly into training, validation, and test sets with probabilities 90%, 5%, and 5%, respectively.

We briefly summarize the baseline algorithms that we compare our methods against.

- **Item KNN**: The method described in Sect. 3 uses the presence of other items in the user interaction history to predict the likelihood of interacting with an item. Multiple different similarity measures can be used for selecting similar items [15], as well as the *shrink* and the *TF-IDF* or *BM25* weighting heuristics [23].
- **$P^3\alpha$ and $R^3\beta$**: Random-walk based methods, proposed in [70,71]. While technically not nearest neighbor algorithms, both of them use closely related ideas.
- **iALS**: Matrix factorization for implicit feedback datasets [72].
- **SLIMElasticNet**: *Sparse Linear Models* is a well-performing regression-based method for *top-n* recommendation [73]. Similar to [12,69], we use the more efficient version introduced in [74].
- **$EASE^R$**: A shallow autoencoder-like model, with a closed form solution for the training objective [75].
- **MultVAE**: A variational autoencoder architecture for collaborative filtering recommendation, introduced in [57]. We chose this one as it performed best against baselines when reproduced by [12,69].

An important recent class of recommender algorithms is based on recurrent neural networks with ideas stemming from natural language processing. Such algorithms include GRU4Rec [30], Transformers4Rec [34], and more [31,35]. However, these methods deal with *sequential* or *session-based* recommendation, as opposed to the classic collaborative filtering model considered in this paper. Since neither our problem setting and datasets nor our algorithms involve sequential context for the items, they cannot be directly compared to recurrent neural network-based methods. We also note that a recent empirical evaluation study [13] finds that the nearest neighbor and other simple approaches remain competitive even for session recommendations, however, such a comparison is beyond the scope of the present paper.

5.3 Inverse frequency scaling

The traditionally used TF-IDF formula and the probabilistic inverse frequency scaling described in Sect. 4.4 fulfill similar roles in the algorithm. They are applied at different steps in the process, however, TF-IDF is a pre-scaling for the data, while the probabilistic version is integrated into the equation that calculates the values for the ensemble formula. Also, different is the fact that TF-IDF scales both based on the predicted and the predicting item (i.e., item and its neighbor),

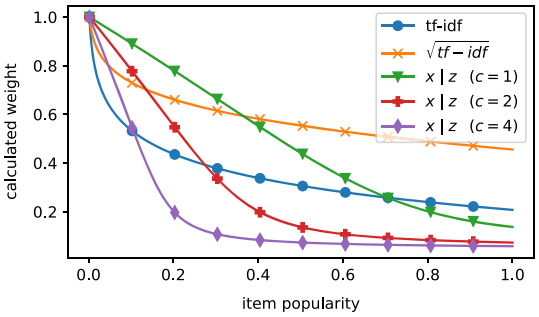

Fig. 1 The relative weight of different frequency scaling schemes, at different rates, all of which discount the popular items

while the probabilistic version only considers the frequency of the predicted item.

Still, we can compare them somewhat directly, because they both scale the weights used for calculating the score of an item, based on its frequency in the dataset. Further complicating the issue is that TF-IDF is used before the similarity function. For example with the cosine similarity, this effectively results in an additional square root in the weighting formula, when considered as a function of the frequency of item i:

$$\begin{aligned}&\frac{\sum_k \text{tf-idf}(x_{ik})\text{tf-idf}(x_{jk})}{\sqrt{\sum_k \text{tf-idf}(x_{ik})}\sqrt{\sum_k \text{tf-idf}(x_{jk})}}\\&= \frac{\text{tf-idf}_i}{\sqrt{\text{tf-idf}_i}}\frac{\sum_k x_{ik}\text{tf-idf}(x_{jk})}{\sqrt{\sum_k x_{ik}}\sqrt{\sum_k \text{tf-idf}(x_{jk})}}.\end{aligned} \tag{43}$$

Note that, as per Sect. 3, variables $x_{i,k}$ are $0-1$ valued, thus the TF-IDF scaling can indeed be treated as a multiplicative coefficient.

Based on the discussion on the interaction of TF-IDF and the similarity function, we plot both the regular TF-IDF weighting and its square root against the probabilistic version on Fig. 1 for the Hetrec training set. Weights are normalized to $f(0) = 1$ to show their relative magnitudes. Popularity is given as a percentage of the most frequent item. Values are also dependent on sample size and number of positive samples, which, for this example, are chosen as 100 and 20, respectively.

The main contribution of Sect. 4.5 is illustrated in Fig. 1. We can observe that TF-IDF starts heavily discounting the weight of items sooner, while the probabilistic version starts off more gradually. On the other hand, the probabilistic version ends up discounting the weights more heavily at higher popularity values. The slope is controlled by hyperparameter c defined in Eq. (22). We observe that the slope of the theoretically justified methods discount for popularity better than heuristic mathods based on TF-IDF.

Table 3 Performance difference using the ensemble formula of Eq. (11) ("or") compared to the classic formula of Eq. (4) ("sum") on the validation sets

	NDCG@50			Recall@50		
	"or"	"sum"	ratio	"or"	"sum"	ratio
Epinions	0.040	0.040	1.004	0.109	0.109	1.002
Amazon Books	0.159	0.156	1.019	0.333	0.329	1.010
Amazon Video	0.191	0.190	1.003	0.391	0.391	1.000
MovieLens 1M	0.270	0.271	0.997	0.439	0.440	0.997
Hetrec	0.220	0.219	1.005	0.354	0.356	0.996

5.4 The ensemble formula

The "or" ensemble formula defined in Eqs. (11) and (21) calculates a prediction close to that of the "sum" ensemble formula (4). However, because we evaluate based on ranking lists, very small differences in score could potentially lead to significant changes in the performance as measured by our metrics. To evaluate the difference between the two possible ensemble formulas, we run experiments with the same settings for both formulas. A bit of caution is needed in the comparison since Eq. (4) carries no guarantees regarding whether the resulting values can be interpreted as probabilities. For this reason, applying the methods introduced in Sect. 4.4 and 4.5 would be questionable. We thus run the experiments using Eqs. (4) and (11), ignoring Eq. (21) and do not handle popularity bias in this comparison experiment.

The comparison of the two ensemble formulas is presented in Table 3. We can observe that the variation in the scores is very small, to the point of being negligible. The measurements confirm our interpretation of the prediction formula as Eq. (11). Without popularity bias mitigation, our new formula performs almost identical to the classical KNN prediction formula that uses the "sum" formula of Eq. (4). This supports the plausibility of our explanations.

5.5 Negative predictors

In Sect. 4.3, we described how we can incorporate an interaction not happening as a predictor into the ensemble formula. When ranking neighbors for each item, we indeed observe that we can find many such predictors with seemingly high enough predictive power to make it to the top-k predictors.

In Fig. 2, we plot the average percent of negative predictors ranked in the top-k for each popularity percentile of items when measured on the Hetrec dataset. We can observe that generally, lower popularity items tend to have more negative neighbors, while the most popular items have none. Note that, due to the power-law distribution of item popularity, the change of popularity is not linear in the percentile. This could explain the sharp drop at around the 50th percentile.

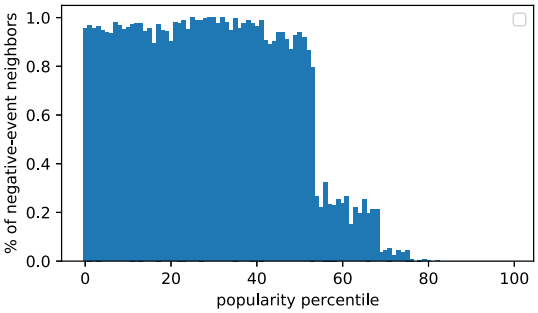

Fig. 2 Average percent of negative neighbors in the top-k for each popularity percentile in the Hetrec dataset. Higher percentiles contain more popular items

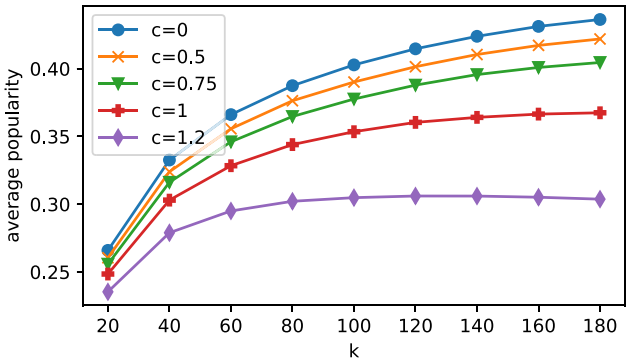

Fig. 3 Change of average popularity of recommended item with different values of k and c on the Hetrec dataset. We can observe that using a larger number of neighbors indeed leads to higher average popularity, as predicted by Eq. (20). Our inference-based method, however, is able to counteract this effect to some degree, as higher values for parameter c result in less popularity increase

Although many negative predictors were estimated to be good predictors, including them in the ensemble formula results in a predictive performance loss of at least 50% of the model according to our experiments. This leads us to believe that despite the high estimated predictive power of these events, they are spurious most of the time.

When we extend the set of possible predictor events to include the ones based on not interacting with items, the number of potential events increases by a very large number. This is due to the fact that most users interact with only a small percentage of items. With the number of possible events increasing, the likeliness of including spurious predictors increases as well. This effect is particularly problematic with less popular items, where the small sample sizes make it harder to find successful predictors and result in higher variance in the measured predictive power. As a result, we end up excluding such events from our model in further experiments.

Regarding the average relative popularity of recommended items, we can observe a large decrease. When measuring item popularity as the number of interacting users, normalized by the maximum of such numbers, the average popularity of recommendations decreases from 0.4263 to 0.2073 when measured at a cutoff of 50 on the HetRec dataset. Such a significant change in itself can be expected to worsen the measured accuracy of the model, as the distribution of the recommendations should approximately reflect the data distribution to achieve best offline evaluation results. Nevertheless, some experiments [13,76] suggest that offline evaluation does not necessarily reflect user satisfaction in such cases.

5.6 Neighborhood size and popularity bias

As we increase the number of neighbors k, we ground the recommendation on more items that each carry a popularity bias itself. The reasoning behind Eq. (20) implies that popularity bias increases with k. To verify, we measure the average popularity of the recommended items while changing k. We also run the experiment with different values of c, the parameter that controls the strength of the popularity bias in Eq. (22). We show the results in Fig. 3. Average popularity on the y axis is measured by a linear scale, with the value 1 for the most and 0 for the least popular item in the data.

With $c = 0$, our algorithm behaves similar to a classic one without any bias reduction. We can observe the average popularity rising with k, which reinforces the correctness of our assumption that the popularity component is considered multiple times in a classical KNN method.

For larger values of c, we quantify our mitigation attempts. Larger values of c clearly reduce the average popularity of the recommendation list. By conflating Fig. 3 with Fig. 5, we can also note that the optimal value of $c \approx 1.25$ seems to cause the average popularity of the recommended items to flatten out to a constant after an initial rise with k.

5.7 Predictive performance

Overall predictive performance is summarized in Table 4 for our algorithm and the baseline KNN methods that we improve upon. We also separately report results for more advanced algorithms. Scores equal to or higher than our method are highlighted in bold. Unfortunately, we are not able to report the performance of the EASE^R algorithm on the Epinions dataset, as the computation would not complete in a reasonable amount of time, even with considerable computing resources, due to the relatively large number of items in this dataset.

We observe that our method improves predictive performance over the KNN baselines in every case when measured by the optimization target NDCG@50, and also in almost all cases when measured by Recall@50. We can observe very similar behavior when measuring on the smaller cutoff of 20 in Table 5.

Table 4 Predictive performance results on the test set with a cutoff of 50

	Amazon Books		Amazon Video		Epinions		Hetrec		MovieLens 1M	
	N@50	R@50	N@50	R@50	N@50	R@50	N@50	R@50	N@50	R@50
Item KNN cosine	0.1785	0.3565	0.1822	**0.3845**	0.0394	0.1044	0.2255	0.3701	0.2724	0.4464
Item KNN dice	0.1785	0.3528	0.1912	**0.3818**	0.0389	0.1046	0.2128	0.3600	0.2569	0.4209
Item KNN jaccard	0.1791	0.3539	0.1906	**0.3819**	0.0388	0.1045	0.2133	0.3499	0.2567	0.4189
Item KNN asymmetric	0.1809	0.3476	0.1966	**0.3987**	0.0427	0.1136	0.2245	0.3749	0.2705	0.4436
Item KNN tversky	0.1820	0.3526	0.1915	0.3808	0.0397	0.1068	0.2156	0.3495	0.2611	0.4253
Our variant	**0.1842**	**0.3620**	**0.2022**	**0.3813**	**0.0451**	**0.1231**	**0.2368**	**0.3886**	**0.2812**	**0.4564**
IALS	0.1287	0.2944	0.1857	0.3766	0.0422	0.1145	0.2360	**0.3990**	0.2672	0.4444
P3α	0.1812	**0.3670**	0.1929	**0.3979**	0.0410	0.1105	0.2235	0.3863	0.2747	0.4564
RP3β	**0.1864**	0.3605	0.1835	**0.3887**	0.0398	0.1051	0.2255	0.3878	**0.2828**	**0.4658**
EASE^R	0.1515	0.3192	0.1815	0.3785	Nan	Nan	0.2115	0.3511	0.2787	0.4517
SLIMElasticNet	**0.1935**	**0.3812**	0.2008	0.3812	0.0444	0.1160	**0.2399**	0.3886	**0.2927**	**0.4654**
MultVAE	0.1097	0.2474	0.1512	0.3468	0.0403	0.1159	0.2110	0.3773	0.2806	**0.4724**

Table 5 Predictive performance results on the test set with a cutoff of 20

	Amazon Books		Amazon Video		Epinions		Hetrec		MovieLens 1M	
	N@20	R@20	N@20	R@20	N@20	R@20	N@20	R@20	N@20	R@20
Item KNN cosine	0.1571	0.2680	0.1617	0.2879	0.0323	0.0688	0.1858	0.2481	0.2187	0.2929
Item KNN dice	0.1586	0.2714	0.1738	**0.3011**	0.0318	0.0690	0.1725	0.2365	0.2051	0.2724
Item KNN jaccard	0.1591	0.2721	0.1732	**0.3006**	0.0318	0.0690	0.1763	0.2379	0.2056	0.2733
Item KNN asymmetric	0.1632	0.2765	0.1782	**0.3130**	0.0351	0.0753	0.1827	0.2478	0.2168	0.2888
Item KNN tversky	0.1635	**0.2780**	0.1741	**0.3001**	0.0325	0.0702	0.1777	0.2373	0.2082	0.2759
Our variant	**0.1635**	**0.2772**	**0.1843**	**0.2972**	**0.0368**	**0.0812**	**0.1951**	**0.2610**	**0.2260**	**0.3012**
IALS	0.1054	0.1954	0.1680	0.2932	0.0342	0.0743	0.1905	**0.2620**	0.2112	0.2847
P3α	0.1586	0.2731	0.1744	**0.3112**	0.0336	0.0731	0.1792	0.2511	0.2177	0.2940
RP3β	**0.1672**	**0.2823**	0.1637	0.2956	0.0327	0.0689	0.1803	0.2495	0.2247	0.3006
EASE^R	0.1300	0.2289	0.1625	0.2896	Nan	Nan	0.1719	0.2291	0.2228	0.2918
SLIMElasticNet	**0.1713**	**0.2899**	**0.1845**	**0.3051**	**0.0369**	0.0782	**0.1983**	**0.2639**	**0.2374**	**0.3091**
MultVAE	0.0909	0.1672	0.1306	0.2498	0.0318	0.0729	0.1685	0.2449	0.2230	**0.3087**

The KNN baseline methods fluctuate with no clear winner. Also note that, the baseline methods themselves include very similar ingredients as in our proposed method, however, based only on heuristic grounds and implemented to varying degrees. Using a number of heuristic elements in the baseline methods have an effect of performance variance from dataset to dataset, while our algorithm performs well in all cases.

Our approach also manages to outperform a number of more advanced methods, including the neural network-based recommender MultVAE that performed best in the comparison in [12,69]. The only algorithm that performs better in multiple data sets is SLIMElasticNet from 2013 [74]. Note that, for our algorithm, hyperparameter selection was performed for the cutoff 50, in which case even SLIMElasticNet only outperforms ours in half of the cases as seen in Table 5.

An interesting case study is the Amazon Video dataset, where our variant consistently wins in the optimization target NDCG and loses in Recall against other KNN algorithms. From this observation, we might conclude that our variant is quite effective at optimizing for NDCG in a trad-off against Recall on this relatively small dataset, possibly due to the high number of hyperparameters involved. However, we do not observe the same effect in other, larger datasets, such as Amazon Books or Epinions.

5.8 Recommendation diversity

Recommendation diversity measures how diverse the recommendation lists are for different users. Diversity can be measured in various ways, such as using the Gini diversity

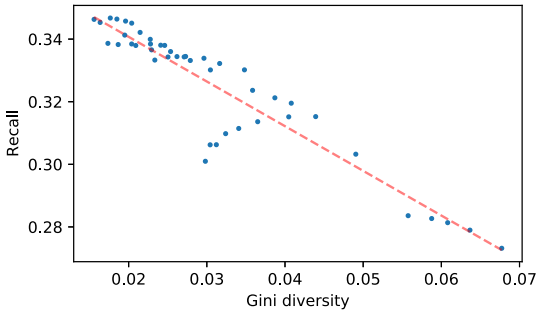

Fig. 4 Change of recall when compared with the Gini diversity of the recommendation lists of different parameter configurations. We also plot the best-fit linear regressor as the red dashed line. We can observe that these two metrics tend to be inversely proportional to each other

measure [77] or the Shannon entropy of the distribution of recommended items over all users.

Several authors [78,79] observed a natural trade-off between diversity and recommendation accuracy. In Fig. 4, we plot the trade-off between recall and Gini diversity by using the data points of Fig. 3. We can clearly observe an inversely proportional relation between the recall and amount of diversity in the ranking lists.

In Table 6, we present the Gini diversity and the Shannon entropy of the recommendation lists of our method and the baselines, at a 50 item cutoff. Our KNN variant achieves a balanced diversity score. It is notable, however, that different scores have very high variance across the evaluated methods. Conflating with Table 4, the accuracy-diversity trade-off is arguably present across methods as well. Still, some methods clearly perform better in both regards then the average, such as the RP3β, which often achieves very high results in both metrics.

While the notion of diversity is related to popularity, our goal was not to improve diversity, but to better model user behavior through explicitly incorporating popularity bias in our model. Accordingly, all of our hyperparameter configurations are optimized for recall, as described in Sect. 5.2. In this paper, we attempt no optimization for a combined measure of accuracy and diversity [78], which could yield different results across methods. To explore whether the presented model can be used for an improved recall-diversity trade-off remains future research.

5.9 Hyperparameters

Next, we measure performance as the function of the various hyperparameters introduced throughout Sect. 4. All newly introduced hyperparameters have the favorable property that a certain minimum or maximum value turns the selected component of the algorithm off: the credible interval lower bound returns the probability itself at $q = 0.5$; the prior parameters a and b have very little effect when they are small; value 0 for the global popularity scaling parameter c denotes no popularity bias, i.e., $Y_i = 0$ when $c = 0$; and finally, α of Eq. (41) has no effect when $\alpha = 1$. With this setup, whenever an optimal parameter value differs from the ones listed above, we can conclude that the corresponding step improves prediction accuracy.

Optimal values vary with the data; we present optimal values in Table 7. For q, optimal values range from 10^{-4} to 0.1. With the prior parameters, we fixed $a = 2.01$ to simplify the optimization process, which still allows us to change the expectation of the prior, i.e., $\frac{a}{a+b}$, by changing the value of b. Note that, we use two separate Beta priors in two steps of our algorithm, in Eq. (8) and in Eq. (32). To distinguish, in

Table 6 Recommended item diversity on the test set for the evaluated methods

	Amazon Books		Amazon Video		Epinions		Hetrec		MovieLens 1M	
	Gini	Entropy	Gini	Entropy	Gini	Entropy	Gini	Entropy	Gini	Entropy
Item KNN cosine	0.4346	14.6999	0.1233	9.3946	0.1209	12.4611	0.0139	7.3247	0.0538	8.0575
Item KNN dice	0.5078	14.7845	0.1785	9.9468	0.0479	10.9559	0.0165	7.4539	0.0498	7.8765
Item KNN jaccard	0.5000	14.7828	0.1776	9.9330	0.0478	10.9496	0.0142	7.2364	0.0498	7.8494
Item KNN asymmetric	0.4686	14.5611	0.1524	9.7078	0.0566	11.5312	0.0158	7.4184	0.0614	8.0926
Item KNN tversky	0.4761	14.6544	0.1789	9.9533	0.0530	11.1388	0.0160	7.3258	0.0530	8.1142
Our variant	0.4431	14.6924	0.1451	9.3348	0.0382	10.3138	0.0139	7.5387	0.0616	8.3081
IALS	0.1472	13.1727	0.1299	9.9664	0.0118	10.8564	0.0208	8.1445	0.0538	8.1307
P3α	0.3073	14.1947	0.1491	9.6472	0.0585	11.6387	0.0147	7.5004	0.0507	8.0022
RP3β	0.4658	14.8078	0.1345	9.4838	0.1446	12.0390	0.0235	7.8296	0.0668	8.2576
EASE^R	0.1577	13.0396	0.1686	9.7731	Nan	Nan	0.0086	6.7291	0.0529	8.1099
SLIMElasticNet	0.2938	14.0509	0.1893	10.2678	0.0344	11.3497	0.0178	7.8192	0.0638	8.3519
MultVAE	0.3637	14.3242	0.2189	10.3845	0.0222	10.7044	0.0250	8.2452	0.0941	8.8399

Diversity is measure by Gini diversity and Shannon entropy, with the recommendation list cutoff of 50

Table 7 Optimal hyperparameters for the datasets

	k	b_1	b_2	c	α	q
Amazon Books	10	11	46	2.63	0.00	0.00100
Amazon Video	100	11	21	2.68	0.00	0.00886
Epinions	757	14	242	4.81	0.17	0.09501
Hetrec	85	45	4	1.20	0.21	0.00016
MovieLens 1M	100	7	11	0.95	0.11	0.00070

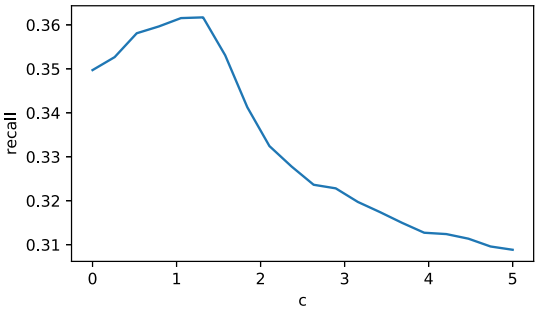

Fig. 5 Change of recall with the hyperparameter c, the strength of the assumed popularity bias to be removed in Eq. (22), over the Hetrec validation set

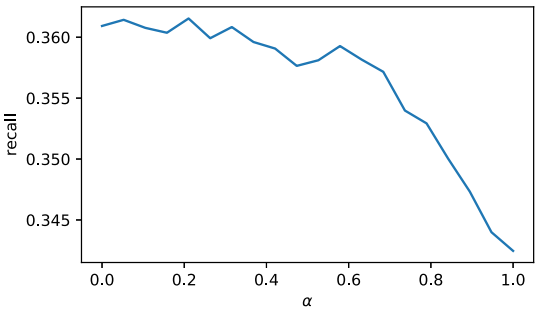

Fig. 6 Change of recall with the hyperparameter α, the weight of popularity after recombining with the popularity-free decoupled estimate, in Eq. (41) on the Hetrec validation set

Table 7, we denote the corresponding parameters b as b_1 and b_2. Optimal values for b_1 were around 10 in most cases, but ranged from 4 to 250 for b_2. Optimal values for c ranged from around 1 to 5, while optimal values for α tend to be near 0 in all cases.

To illustrate the behavior of the method with different values of c and α, we plot the performance in recall with different values of these parameters in Figs. 5 and 6. We observe that the best value for c is slightly above 1, which means that in Eq. (22), we give an even higher probability estimate to the effect of popularity on the user interacting with the given item than what we would infer from the fraction of users consuming the item. Furthermore, best performance is achieved with a low value of α, which corresponds to little weight to popularity after reintroducing it in Eq. (41). Both of these findings confirm the importance of decoupling popularity from taste and the power of the two main constants in Eqs. (22) and (41) that control the strength of the effect.

6 Conclusions

In this work, we addressed the task of implicit feedback recommendation using item-based nearest neighbor methods. We reviewed the classic method and developed a probabilistic explanation of its inner mechanics. We proposed a new interpretation of the prediction process by considering similarity as a conditional probability and proposed incorporating the uncertainty of the observed similarities by using Bayesian credible intervals. Further, we presented an interpretation of the classic ensemble prediction formula that considers the prediction a logical operator over multiple independent predictors. Finally, we developed a novel way of modeling user behavior as a combination of popularity and personal taste and used Bayesian inference to extract the personal taste component. We tested our claims extensively over multiple datasets to assess their validity.

The overall goal of this paper was to explore interpretations of the classic method in a probabilistic context, possibly influencing or inspiring future methods, and to gain an increased understanding of and insight into the mechanics behind these methods. As a result, our proposed solutions also translated into an improvement in recommendation accuracy. In future work, our new algorithm can be evaluated in an ensemble combined with sequential recommenders based on for example recurrent neural networks to complement them with modeling the longer term user taste for recommendation.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s41060-022-00367-4.

Author Contributions DK and AB contributed to conceptualization, methodology, interpretation, manuscript preparation; DK contributed to software.

Funding Open access funding provided by ELKH Institute for Computer Science and Control. Support by the the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory Grant no RRF-2.3.1-21-2022-00004.

Availability of code, data and materials We make our implementations and the code used to run the experiments in this paper, including preprocessing and the exact data splits used, publicly available at https://github.com/proto-n/probabilistic_knn.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethics approval Not applicable.

Consent for publication Not applicable.

References

1. O'Brien, H.L., Toms, E.G.: What is user engagement? A conceptual framework for defining user engagement with technology. J. Am. Soc. Inform. Sci. Technol. **59**(6), 938–955 (2008)
2. Zhang, Z., Wang, B.: Learning sequential and general interests via a joint neural model for session-based recommendation. Neurocomputing **415**, 165–173 (2020)
3. Gwadabe, T.R., Liu, Y.: Improving graph neural network for session-based recommendation system via non-sequential interactions. Neurocomputing **468**, 111–122 (2022)
4. Koenigstein, N., Koren, Y.: Towards scalable and accurate item-oriented recommendations. In: Proceedings of the 7th ACM Conference on Recommender Systems, pp. 419–422 (2013)
5. Pilászy, I., Serény, A., Dózsa, G., Hidasi, B., Sári, A., Gub, J.: Neighbor methods vs. matrix factorization-case studies of real-life recommendations. In: Proceedings of the 9th ACM conference on Recommender systems, vol. 15 (2015)
6. Bennett, J., Lanning, S.: The netflix prize. In: Proceedings of KDD Cup and Workshop, vol. 2007, p. 35 (2007)
7. Liu, Y., Kou, Z.: Predicting who rated what in large-scale datasets. ACM SIGKDD Explor. Newsl. **9**(2), 62–65 (2007)
8. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005)
9. Wang, S., Cao, L., Wang, Y., Sheng, Q.Z., Orgun, M.A., Lian, D.: A survey on session-based recommender systems. ACM Comput. Surv. **54**(7), 1–38 (2021)
10. Song, W., Wang, S., Wang, Y., Wang, S.: Next-item recommendations in short sessions. In: Fifteenth ACM Conference on Recommender Systems (2021)
11. Feng, L., Wei, H., Guo, Q., Lin, Z., An, B.: Embedding-augmented generalized matrix factorization for recommendation with implicit feedback. IEEE Intell. Syst. **36**(6), 32–41 (2021)
12. Dacrema, M.F., Boglio, S., Cremonesi, P., Jannach, D.: A troubling analysis of reproducibility and progress in recommender systems research. ACM Trans. Inf. Syst. (TOIS) **39**(2), 1–49 (2021)
13. Ludewig, M., Mauro, N., Latifi, S., Jannach, D.: Empirical analysis of session-based recommendation algorithms. User Model. User-Adapt. Interact. **31**(1), 149–181 (2021)
14. Bell, R.M., Koren, Y.: Improved neighborhood-based collaborative filtering. In: KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 7–14 (2007)
15. Verstrepen, K., Goethals, B.: Unifying nearest neighbors collaborative filtering. In: Proceedings of the 8th ACM Conference on Recommender Systems, pp. 177–184 (2014)
16. Kalloori, S., Ricci, F., Tkalcic, M.: Pairwise preferences based matrix factorization and nearest neighbor recommendation techniques. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 143–146 (2016)
17. Khawar, F., Zhang, N.L.: Cleaned similarity for better memory-based recommenders. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1193–1196 (2019)
18. Qamar, A.M., Gaussier, E., Chevallet, J.-P., Lim, J.H.: Similarity learning for nearest neighbor classification. In: 2008 Eighth IEEE International Conference on Data Mining (2008)
19. Holmes, C.C., Adams, N.M.: A probabilistic nearest neighbour method for statistical pattern recognition. J. R. Stat. Soc.: Ser. B (Stat. Methodol.) **64**(2), 295–306 (2002)
20. Guo, R., Chakraborty, S.: Bayesian adaptive nearest neighbor. Stat. Anal. Data Min. **3**(2), 92–105 (2010)
21. Cañamares, R., Castells, P.: A probabilistic reformulation of memory-based collaborative filtering: Implications on popularity biases. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 215–224 (2017)
22. Karypis, G.: Evaluation of item-based top-n recommendation algorithms. In: Proceedings of the Tenth International Conference on Information and Knowledge Management, pp. 247–254 (2001)
23. Wang, J., Robertson, S., de Vries, A.P., Reinders, M.J.: Probabilistic relevance ranking for collaborative filtering. Inf. Retr. **11**(6), 477–497 (2008)
24. Park, Y., Park, S., Jung, W., Lee, S.-G.: Reversed CF: a fast collaborative filtering algorithm using a k-nearest neighbor graph. Expert Syst. Appl. **42**(8), 4022–4028 (2015)
25. Ludewig, M., Kamehkhosh, I., Landia, N., Jannach, D.: Effective nearest-neighbor music recommendations. In: Proceedings of the ACM Recommender Systems Challenge 2018 (2018)
26. Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 1–35. Springer, Boston (2011)
27. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. **7**(1), 76–80 (2003)
28. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web, pp. 285–295 (2001)
29. Lops, P., De Gemmis, M., Semeraro, G.: Content-based recommender systems: state of the art and trends. In: Recommender Systems Handbook, pp. 73–105 (2011)
30. 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, pp. 843–852 (2018)
31. 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, pp. 495–503 (2017)
32. 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, pp. 173–182 (2017)
33. Lin, J.: The neural hype and comparisons against weak baselines. In: ACM SIGIR Forum, vol. 52. ACM, New York (2019)
34. de Souza Pereira Moreira, G., Rabhi, S., Lee, J.M., Ak, R., Oldridge, E.: Transformers4Rec: Bridging the Gap between NLP

and Sequential/Session-Based Recommendation, pp. 143–153 (2021)
35. Devooght, R., Bersini, H.: Long and short-term recommendations with recurrent neural networks. In: Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, pp. 13–21 (2017)
36. Guo, W., Wang, S., Lu, W., Wu, H., Zhang, Q., Shao, Z.: Sequential dependency enhanced graph neural networks for session-based recommendations. In: 2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA), pp. 1–10 (2021)
37. Zhang, Z., Wang, B.: Fusion of latent categorical prediction and sequential prediction for session-based recommendation. Inf. Sci. **569**, 125–137 (2021)
38. Zhang, Z., Wang, B.: Graph neighborhood routing and random walk for session-based recommendation. In: 2021 IEEE International Conference on Data Mining (ICDM), pp. 1517–1522 (2021)
39. Frigó, E., Kocsis, L.: Online convex combination of ranking models. User Model User-Adapt. Interact. 1–35 (2021)
40. Adhikari, V.K., Guo, Y., Hao, F., Varvello, M., Hilt, V., Steiner, M., Zhang, Z.-L.: Unreeling netflix: understanding and improving multi-CDN movie delivery. In: 2012 Proceedings IEEE INFOCOM, pp. 1620–1628 (2012)
41. Tran, Q., Tran, L., Hai, L.C., Van Linh, N., Than, K.: From implicit to explicit feedback: a deep neural network for modeling sequential behaviours and long-short term preferences of online users. Neurocomputing **479**, 89–105 (2022)
42. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pp. 452–461 (2009)
43. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, pp. 175–186 (1994)
44. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 43–52 (1998)
45. Ning, X., Desrosiers, C., Karypis, G.: A comprehensive survey of neighborhood-based recommendation methods. In: Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 37–76. Springer, Berlin (2015)
46. Aiolli, F.: Efficient top-n recommendation for very large scale binary rated datasets. In: Proceedings of the 7th ACM Conference on Recommender Systems, pp. 273–280 (2013)
47. Wang, S., Cao, L.: Inferring implicit rules by learning explicit and hidden item dependency. IEEE Trans. Syst. Man Cybern. Syst. **50**(3), 935–946 (2020)
48. Defazio, A.J., Caetano, T.S.: A graphical model formulation of collaborative filtering neighbourhood methods with fast maximum entropy training. In: Proceedings of the 29th International Conference on International Conference on Machine Learning, pp. 555–562 (2012)
49. Pennock, D.M., Horvitz, E., Lawrence, S., Giles, C.L.: Collaborative filtering by personality diagnosis: a hybrid memory- and model-based approach. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, pp. 473–480 (2000)
50. Sovilj, D., Sanner, S., Soh, H., Li, H.: Collaborative filtering with behavioral models. In: Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization, pp. 91–99 (2018)
51. Marlin, B.M., Zemel, R.S.: Collaborative prediction and ranking with non-random missing data. In: Proceedings of the Third ACM Conference on Recommender Systems, pp. 5–12 (2009)
52. Schnabel, T., Swaminathan, A., Singh, A., Chandak, N., Joachims, T.: Recommendations as treatments: debiasing learning and evaluation. In: International Conference on Machine Learning, pp. 1670–1679. PMLR (2016)
53. Steck, H.: Training and testing of recommender systems on data missing not at random. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 713–722 (2010)
54. Steck, H.: Item popularity and recommendation accuracy. In: Proceedings of the Fifth ACM Conference on Recommender Systems, pp. 125–132 (2011)
55. Steck, H.: Evaluation of recommendations: rating-prediction and ranking. In: Proceedings of the 7th ACM Conference on Recommender Systems, pp. 213–220 (2013)
56. Yang, L., Cui, Y., Xuan, Y., Wang, C., Belongie, S., Estrin, D.: Unbiased offline recommender evaluation for missing-not-at-random implicit feedback. In: Proceedings of the 12th ACM Conference on Recommender Systems, pp. 279–287 (2018)
57. Liang, D., Krishnan, R.G., Hoffman, M.D., Jebara, T.: Variational autoencoders for collaborative filtering. In: Proceedings of the 2018 World Wide Web Conference, pp. 689–698 (2018)
58. Shani, G., Gunawardana, A.: Evaluating recommendation systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 257–297. Springer, Boston (2011)
59. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM Trans. Inf. Syst. (TOIS) **22**(1), 143–177 (2004)
60. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ, Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P.: SciPy 1.0 contributors: SciPy 1.0: fundamental algorithms for scientific computing in python. Nat. Methods **17**, 261–272 (2020)
61. Zheng, L., Lu, C.-T., Jiang, F., Zhang, J., Yu, P.S.: Spectral collaborative filtering. In: Proceedings of the 12th ACM Conference on Recommender Systems, pp. 311–319 (2018)
62. McAuley, J., Targett, C., Shi, Q., van den Hengel, A.: Image-based recommendations on styles and substitutes. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 43–52 (2015)
63. Harper, F.M., Konstan, J.A.: The movielens datasets: history and context. ACM Trans. Interact. Intell. Syst. **5**(4), 1–19 (2015)
64. Cantador, I., Brusilovsky, P., Kuflik, T.: Second workshop on information heterogeneity and fusion in recommender systems (hetrec2011). In: Proceedings of the Fifth ACM Conference on Recommender Systems, pp. 387–388 (2011)
65. Massa, P., Avesani, P.: Trust-aware recommender systems. In: Proceedings of the 1st ACM Conference on Recommender Systems, pp. 17–24 (2007)
66. He, R., McAuley, J.: Ups and downs: modeling the visual evolution of fashion trends with one-class collaborative filtering. In: Proceedings of the 25th International Conference on World Wide Web, pp. 507–517 (2016)
67. Batagelj, V., Zaveršnik, M.: Fast algorithms for determining (generalized) core groups in social networks. Adv. Data Anal. Classif. **5**(2), 129–145 (2011)
68. Krichene, W., Rendle, S.: On sampled metrics for item recommendation. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1748–1757 (2020)
69. Ferrari Dacrema, M., Cremonesi, P., Jannach, D.: Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In: Proceedings of the 13th ACM Conference on Recommender Systems, pp. 101–109 (2019)
70. Cooper, C., Lee, S.H., Radzik, T., Siantos, Y.: Random walks in recommender systems: Exact computation and simulations. In:

Proceedings of the 23rd International Conference on World Wide Web, pp. 811–816 (2014)
71. Paudel, B., Christoffel, F., Newell, C., Bernstein, A.: Updatable, accurate, diverse, and scalable recommendations for interactive applications. ACM Trans. Interact. Intell. Syst. **7**(1), 1–34 (2016)
72. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: 2008 Eighth IEEE International Conference on Data Mining (2008)
73. Ning, X., Karypis, G.: Slim: Sparse linear methods for top-n recommender systems. In: 2011 IEEE 11th International Conference on Data Mining, pp. 497–506 (2011)
74. Levy, M., Jack, K.: Efficient top-n recommendation by linear regression. In: RecSys Large Scale Recommender Systems Workshop (2013)
75. Steck, H.: Embarrassingly shallow autoencoders for sparse data. In: The World Wide Web Conference, pp. 3251–3257 (2019)
76. Ludewig, M., Jannach, D.: User-centric evaluation of session-based recommendations for an automated radio station. In: Proceedings of the 13th ACM Conference on Recommender Systems, pp. 516–520 (2019)
77. Adomavicius, G., Kwon, Y.: Improving aggregate recommendation diversity using ranking-based techniques. IEEE Trans. Knowl. Data Eng. **24**(5), 896–911 (2011)
78. Raza, S., Ding, C.: A regularized model to trade-off between accuracy and diversity in a news recommender system. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 551–560 (2020)
79. Raza, S., Ding, C.: Deep neural network to tradeoff between accuracy and diversity in a news recommender system. In: 2021 IEEE International Conference on Big Data (Big Data), pp. 5246–5256 (2021)