Expert Systems With Applications 115 (2019) 535–542

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Recommender system based on pairwise association rules

Timur Osadchiy[a,*], Ivan Poliakov[a], Patrick Olivier[a], Maisie Rowland[b], Emma Foster[b]

[a] Open Lab, School of Computing, Newcastle University, Newcastle upon Tyne, United Kingdom
[b] Institute of Health and Society, Newcastle University, Newcastle upon Tyne, United Kingdom

ARTICLE INFO

Article history:
Received 4 April 2018
Revised 9 July 2018
Accepted 10 July 2018
Available online 21 August 2018

Keywords:
Association rules
Cold-start problem
Data mining
Ontologies
Recommender systems

ABSTRACT

Recommender systems based on methods such as collaborative and content-based filtering rely on extensive user profiles and item descriptors as well as on an extensive history of user preferences. Such methods face a number of challenges; including the cold-start problem in systems characterized by irregular usage, privacy concerns, and contexts where the range of indicators representing user interests is limited. We describe a recommender algorithm that builds a model of collective preferences independently of personal user interests and does not require a complex system of ratings. The performance of the algorithm is analyzed on a large transactional data set generated by a real-world dietary intake recall system.

1. Introduction

Recommender systems aim to identify consumer preferences and accurately suggest relevant items (e.g. products, services, content). They are used in various application domains, including online retail, tourism and entertainment (Covington, Adams, & Sargin, 2016; Linden, Smith, & York, 2003). Widely adopted recommendation techniques often utilize collaborative filtering or content-based recommendation methods (Pazzani & Billsus, 2007).

Collaborative filtering produces recommendations based on user preference models that are generated from explicit and/or implicit characteristics and metrics corresponding to user interests. Explicit indicators normally imply users assigning ratings to items; for example, to products viewed in, or purchased from, an online store. Examples of implicit indicators include the amount of time users spend interacting with content (e.g. watching a video) or levels of interaction (e.g. scroll offset of a web page containing an article they are reading). Items that are positively rated or purchased by consumers with similar preference models are used as recommendations for target users. User similarity can be expressed through correlations in purchasing history or ratings given to the same products, which can be further amplified with demographics (e.g. age, gender, occupation). Content-based filtering identifies similarities between items based on a set of their descriptors (e.g. purpose of an item, author, artist, keywords). Items similar to those positively rated or purchased by the target user, are used as recommendations.

Collaborative and content-based filtering recommender systems are therefore heavily dependent on extensive user- or item- profile information and are most effective when there is a rich history of user preferences or behavior. Sparse data sets and lean user profiles typically result in low-quality recommendations or an inability to produce recommendations at all. This is referred to as the cold-start problem, where new users are added into the system with empty behavior profiles or new items are added that have not been reviewed or rated by anyone (Shaw, Xu, & Geva, 2010). Many solutions to the cold-start problem have been considered, including hybrid methods that combine collaborative and content-based filtering (Schein, Popescul, Ungar, & Pennock, 2002), and methods that aim to predict user preferences from demographics (Lika, Kolomvatsos, & Hadjiefthymiades, 2014), or knowledge of social relationships (Carrer-Neto, Hernández-Alcaraz, Valencia-García, & García-Sánchez, 2012).

There are, however, a number of application contexts where users interact anonymously; for example, online shops where an unregistered user browses and adds products to their basket to check out later. In other contexts, such as email recipient recommendation (Roth et al., 2010), applications requiring high levels of privacy, or those where individual interactions with a system are necessarily infrequent (e.g. dietary surveys Bradley et al., 2016) there are no features and ratings to exploit, and the construction

* Corresponding author at: Open Lab, School of Computing, Newcastle University, Urban Sciences Building, 1 Science Square, Science Central, Newcastle upon Tyne, NE4 5TG, United Kingdom.
E-mail addresses: t.osadchiy@newcastle.ac.uk (T. Osadchiy), ivan.poliakov@newcastle.ac.uk (I. Poliakov), patrick.olivier@newcastle.ac.uk (P. Olivier), maisie.rowland@newcastle.ac.uk (M. Rowland), emma.foster@newcastle.ac.uk (E. Foster).

https://doi.org/10.1016/j.eswa.2018.07.077

of personal behavior and preference models is not possible. To address these challenges we describe a recommender algorithm that is independent of any personal user model and does not require a complex system of ratings. Based on a set of observed items selected by a user, the algorithm produces a set of items ranked by confidence of their being observed next. In designing the underlying algorithm, we review existing methods that aim to address similar tasks, adapt them to meet the constraints of the application context that is our primary concern (dietary surveys), and propose a novel alternative. The performance of three methods is compared through the task of recommending omitted foods in a real world dietary recall system.

2. Related work

While various approaches have been proposed to address the cold-start problem in recommender systems, the majority of these rely on knowledge of content (Popescul, Pennock, & Lawrence, 2001) and users (Lika et al., 2014), including social relationships between users (Carrer-Neto et al., 2012), whereas our concern is with contexts where such information is not available. Shaw et al. addressed the cold-start problem in recommender systems by using a data analysis technique, which is applied to large data sets for discovering items that frequently appear together in a single transaction. This technique is known as association rules (Agrawal, Imielinski, & Swami, 1993; Shaw et al., 2010). Each association rule normally consists of a set of antecedent items that lead to a consequent item with a certain confidence. Pazzani and Billsus consider the list of topics of books users voted for as transactions, which allowed them to extract association rules for topics that frequently appeared together as part of a user's interests (Pazzani & Billsus, 2007). To expand preferences for each user, the algorithm then generates all possible combinations of topics for every book the user voted for and filters association rules, where the antecedent part of the rule matches one of the combinations. The consequent list of topics is added to the preferences of that user.

In combination with a domain ontology association rules can be effectively employed for extracting, understanding and formalizing new knowledge (Ruiz, Foguem, & Grabot, 2014; Sene, Kamsu-Foguem, & Rumeau, 2018). However, association rules have to be adapted for recommendation tasks since they are primarily designed to be used as exploratory tools (Rudin, Letham, Salleb-Aouissi, Kogan, & Madigan, 2011) to discover previously unknown relations that need to be analyzed for their interestingness (Atkinson, Figueroa, & Pérez, 2013). As we will probably want to provide more specificity, and recommend the exact titles of the books instead of generic categories, this potentially leads to a vast number of mined association rules and matching all possible combinations of the observed items may not result in rules being found. Furthermore, a consequent item may appear in multiple matching rules, meaning that a function must be introduced that aggregates the confidences of found rules into a single score for the consequent item. Finally, only the associations with a support (i.e. how often a rule holds as true across the data set) higher than a defined threshold are normally extracted (Li & Deng, 2007). The produced list of rules is supposed to be of a reasonable size, to allow manual examination. In a recommender system, even associations with low frequencies could still be relevant, if other relevant rules with higher confidence are not found. This requires the extraction of as many rules as possible, making the mining process a computationally expensive task (Zheng, Kohavi, & Mason, 2001).

Roth et al. (2010) introduced a method for building implicit social graphs based on histories of interaction between users and estimations of their affinity and applied it to the problem of email recipient recommendation. Based on a set of email addresses selected by a user (the seed group) the algorithm extracts all groups of contacts with whom the user has ever exchanged emails. Here a group of contacts is a set of email contacts that were observed together in a recipient list. For each of the contacts in each group, excluding members of the seed group, an interaction score is calculated based on the volume of messages exchanged with that group, the recency of those messages, and the number of intersections of the seed group with the group of contacts that is being considered. Interaction scores of contacts that are present in multiple groups are aggregated into a single interaction score, which is then used to rank the set of email recommendations.

The implicit social graph is a promising alternative to mining association rules. It instead measures the confidence of recommended items based solely on observed transactions that are prefiltered by intersections with given items. However, the method also estimates the relevance of a group of recommended emails based on the strength of social interaction of the target user with that group, which is not a meaningful metric for applications that do not assume communication (or other interaction) between users.

Association rules and implicit social graphs are data entities that represent item-to-group relationships. However, DuMouchel and Pregibon (2001) suggested that a more efficient approach to discovering "interesting" associations is to first find pairs of items that frequently appear together and then analyze larger sized item sets that contain those pairs. For example, if *ABC* appears in a data set with a certain frequency, then pairs *AB, BC* and *AC* would be at least as frequent as the triplet. Raeder and Chawla (2011) effectively analyzed associations through a graph of individual items connected to each other with edges that are weighted by the frequency of two items appearing together. Items that have stronger relationships with each other are compared to other items form clusters, which are then targeted for further analysis. Similar to the implicit social graph this method avoids the need to mine all possible association rules, but without requiring any additional indicators of relevance except for the item pairs frequencies. The relevance of produced recommendations is effectively inferred from the likelihood of their appearing with the observed items.

3. Associated food recommender algorithm

3.1. Intake24

We introduce a new recommendation algorithm that was developed for Intake24, a system for conducting 24-h multiple-pass dietary recall surveys (Bradley et al., 2016). Intake24 is designed to be a cost-effective alternative to interviewer-led 24-h recalls and provides respondents with a web-based interface through which they enter their dietary intake for the previous day. Respondents will likely only ever use the system if they are a part of a dietary study and only for a short period of time.

Within a survey, a respondent typically records their dietary intake for the previous day on three separate occasions. A single day normally consists of four to seven meals (e.g. breakfast, morning snack, lunch etc.) which include a selection of foods, drinks, desserts, condiments, and such (referred to generically as foods). During the first step of a recall session, a respondent reports a list of names of foods consumed during each of the previous day's meals in a free-text format. For each text entry, the system returns a list of relevant foods selected from a taxonomy of around 4800 foods, organized in a tree-based, multi-level structure. Specific foods are terminal nodes of this taxonomy and are linked to their nutrient values and portion size estimation methods. Respondents select one food from the returned list to add to their meal; for example: *Coca-Cola (not diet); Beef, stir fried (meat only); Tomatoes, tinned; Basmati rice; Onions, fried; Chilli powder; Kidney beans.*

Completing an accurate recall requires respondents to be able to identify foods they ate from a database that covers more than 4800 foods; for example, there are more than 30 types of bread alone. Thus, one of the key features in determining the usability of a dietary recall system is its presentation of food search results. If respondents are not able to readily identify items from a list returned in response to their textual description of the food, they are more likely to select foods perceived as the closest match or even skip reporting the intake of that food. In other words, the relevance of search results, in terms of prioritizing them appropriately, may directly affect the accuracy of dietary recall through level of effort and time required to select the correct foods and report intake.

The main application for the recommender algorithm is to automate the extraction of questions about foods that are commonly consumed together (associated foods). In Intake24, this feature is implemented as a link between an antecedent food (e.g. *toast, white bread*) and the consequent associated food category (e.g. *butter/margarine/oils*) along with a question that is asked if a respondent selects the antecedent food (e.g. *Did you have any butter or margarine on your toast?*). Such food associations prompts are currently hand-crafted by trained nutritional experts, which for thousands of foods is inevitably a time consuming process that is prone to omissions. Eating habits depend on region, culture, diet, and a number of other factors, which requires defining new associations for every context in which a system is deployed. Furthermore, over time new foods and recipes emerge and dietary trends change. Indeed, existing rules are often curated based only on personal experience or previous research, and no published study has evaluated their appropriateness or explored alternative data driven approaches to extracting such associations.

3.2. Generic procedure

Our approach assumes that the patterns in eating behaviors of an observed population; that is, the respondents who took part in surveys conducted in a given country, has some relevance to those of an individual in that population. The recommender algorithm assumes no prior knowledge about an individual except their currently selected food items. The algorithm is trained on a large set of observed meals and produces a model of the eating behavior of a given population, where a meal is a group of uniquely identifiable foods (e.g. vanilla ice cream, pear juice) reported to be eaten on a single occasion. Each individual food can be recorded as being eaten only once during a meal. During the recommendation step, the resulting model accepts a set of foods, which we refer to as input foods *IF*, and returns a set of recommended foods *RF* mapped to likelihoods of being reported along with *IF* (recommendation scores). *IF* are excluded from recommendations. In the following section we discuss three possible implementations that were considered for the recommender algorithm. Along with the description of our methods we include examples of generated models and recommendations for a sample transaction data set.

3.3. Association rules

We introduce a recommender algorithm based on association rules (AR) that generates a model of eating behavior from a data set of meals (in the training step) in the form of association rules. Each rule consists of a set of antecedent foods and a single consequent food, together with the confidence that the consequent food will be present in a meal given the antecedent foods that were observed. The procedure for retrieving association rules is described in Agrawal et al. (1993).

The AR algorithm makes predictions from stored association rules with antecedent part *antc* similar to *IF*, and produces recommendations from the consequent parts of the rules. To do so, AR takes association rules that have a consequent food that is different from any of *IF* and the antecedent foods *antc* that include at least one of *IF* (Algorithm 1). The algorithm calculates the likelihood of a recommended food f to be selected next as the confidence of the rule c multiplied by the similarity between *antc* and *IF* (i.e. match score *ms*). The match score *ms* is calculated as the number of foods that appear both in *IF* and *antc* (i.e. intersections) raised to the power of two and divided by the size of *IF* and the size of *antc*. We introduce the match score so that the recommendations from the rules with *antc* that are more similar to *IF* produce recommendations that will appear higher. We then sum the scores for every f as its single recommendation score $RF[f]$.

Algorithm 1: Recommendations based on association rules.

```
function Recommend
    input  : AM, association rules based model
             IF, foods selected by a respondent
    returns: RF, list of food recommendations
1   RF ← ∅;
2   foreach rule rl ∈ AM & rl.consequent ∉ IF :
3       f ← rl.consequent;
4       if ∃ af : af ∈ rl.antecedent & af ∈ IF :
5           if f ∉ RF :
6               RF[f] ← 0
7           antc ← rl.antecedent;
8           c ← rl.confidence;
9           intr ← size({af : af ∈ antc & af ∈ IF});
10          ms ← intr^2 / (size(antc) * size(IF));
11          RF[f] ← RF[f] + c * ms
12  return RF
```

Recommendations produced by AR applied to the example transaction data set {*abcd, ade, de, ab*} and given items {*ab*} are provided in Table 1.

3.4. Transactional item confidence

We adapted the implicit social graph method described in Roth et al. (2010) for our food recommendation task, which resulted in a recommender algorithm based on transactional item confidence (TIC). One key difference to our food recommendation problem is that the original email recipient recommendation task for which the implicit social graph was developed assumed two types of relationships between items in a data set (outgoing and incoming emails). Our data set assumes only one type of relationship, which is the co-occurrence of foods in a meal. For that reason, TIC produces recommendations based on similarity of historically observed transactions to *IF* and the frequency of foods appearing in those transactions.

During the training step, TIC converts all reported meals to a map of unique meals (or transactions) *TM*, so that there are no two transactions of the same length containing the same foods (Algorithm 2). For every food f in a transaction m, the confidence (conditional probability) is calculated as $TM[m, f]$ of f being present in m given that the rest of the foods from m were observed. To do so, the algorithm counts the number *cf* of reported meals that contain all the foods from m, excluding f, and divides it by the number *cm* of reported meals containing all of the foods from m. This is similar to the confidence measured in AR, but in this case we calculate it only for the full-sized meals that were observed in the data set M, and not for all possible combinations of foods within those meals.

At the recommend step, the algorithm retrieves all transactions containing any of the input foods *IF* (Algorithm 3). Within each of

Table 1
Recommender algorithm based on association rules applied to the example data set.

Model based on AR	Filtered rules	Recommendations
1. a ⇒ b 0.67, d 0.67, c 0.33, e 0.33	1. **a** ⇒ d 0.67, c 0.33, e 0.33	d: 2.50
2. b ⇒ a 1.00, c 0.50, d 0.50	2. **b** ⇒ c 0.50, d 0.50	c: 1.96
3. c ⇒ a 1.00, b 1.00, d 1.00	6. **a, b** ⇒ c 0.50, d 0.50	e: 0.29
4. d ⇒ a 0.67, e 0.67, b 0.33, c 0.33	7. **a**, c ⇒ d 1.00	
5. e ⇒ d 1.00, a 0.50	8. **a**, d ⇒ c 0.50, e 0.50	
6. a, b ⇒ c 0.50, d 0.50	9. **a**, e ⇒ d 1.00	
7. a, c ⇒ b 1.00, d 1.00	10. **b**, c ⇒ d 1.00	
8. a, d ⇒ b 0.50, c 0.50, e 0.50	11. **b**, d ⇒ c 1.00	
9. a, e ⇒ d 1.00	14. **a, b**, c ⇒ d 1.00	
10. b, c ⇒ a 1.00, d 1.00	15. **a, b**, d ⇒ c 1.00	
11. b, d ⇒ a 1.00, c 1.00		
12. c, d ⇒ a 1.00, b 1.00		
13. d, e ⇒ a 0.50		
14. a, b, c ⇒ d 1.00		
15. a, b, d ⇒ c 1.00		
16. a, c, d ⇒ b 1.00		
17. b, c, d ⇒ a 1.00		

Algorithm 2: Training the model based on transactional item confidence.

```
function Train
    input  : M, data set of all meals
    returns: TM, map of unique meals with confidence for
             every food
TM ← ∅;
foreach meal m ∈ M :
    if m ∉ TM :
        TM[m] ← ∅
    cm ← size({m1 : m1 ∈ M & m ∈ m1});
    foreach food f ∈ m :
        m2 ← {f1 : f1 ∈ m & f1 ≠ f};
        cf ← size({m3 : m3 ∈ M & m2 ∈ m3});
        TM[m, f] ← cf/cm
return TM
```

Algorithm 3: Recommendations based on transactional item confidence.

```
function Recommend
    input  : TM, map of unique meals with confidence for
             every food
             IF, foods selected by a respondent
    returns: RF, list of food recommendations
RF ← ∅;
foreach meal m ∈ TM :
    if ∃ f1 : f1 ∈ m & f1 ∈ IF :
        foreach food f ∈ m & f ∉ IF :
            if f ∉ RF :
                RF[f] ← 0
            conf ← m[f];
            inter ← size({f2 : f2 ∈ m & f2 ∈ IF});
            RF[f] ← RF[f] + inter ∗ conf
return RF
```

the retrieved transactions m, foods f that are not included in IF are mapped to a score that is calculated as the number of intersections of m with IF (i.e. similarity) multiplied by the food's confidence $TM[m, f]$. Multiple scores for f measured from different transactions are summed into a final recommendation score $RF[f]$.

Recommendations produced by the TIC applied to an example transaction data set {*abcd, ade, de, ab*} given items {*ab*} are provided below (Table 2).

3.5. Pairwise association rules

Unlike the previous two algorithms, which produce recommendations from association rules and transactions similar to currently observed *IF*, the recommender algorithm based on pairwise association rules (PAR) recommends foods that are likely to be observed with any of *IF* in pairs. During the training stage (Algorithm 4), PAR for every observed food f counts the number $OD[f]$ of meals

Algorithm 4: Training the model based on pairwise association rules.

```
function Train
    input  : M, data set of all meals
    returns: PM, pairwise association rules
OD ← ∅, food occurrences;
CD ← ∅, food co-occurrences;
foreach meal m ∈ M :
    foreach food f ∈ m :
        if f ∉ OD & f ∉ CD :
            OD[f] ← 0;
            CD[f] ← ∅;
        OD[f] ← OD[f] + 1;
        foreach food f1 ∈ m & f1 ≠ f :
            if f1 ∉ CD[f] :
                CD[f, f1] ← 0
            CD[f, f1] ← CD[f, f1] + 1
PM ← [OD, CD];
return PM
```

that contain that food. For every observed pair of foods $\{f, f1\}$, it also counts the number $CD[f, f1]$ of reported meals that contain that pair. At the recommend step (Algorithm 5), PAR retrieves pairs $CD[inf]$, where one food inf is observed in IF. For every pair $\{inf, f\}$, the algorithm calculates the conditional probability p, of f being in a meal, given that inf was observed as the number of meals that contain that pair $CD[inf, f]$, divided by the number of meals $OD[inf]$ that contain only inf. For example, if item A appeared 10 times in the data set and co-occurred with item B only 2 times, then the conditional probability that item B will occur the next time the A is present is 0.2. Multiple probabilities retrieved for f from differ-

Table 2
Recommender algorithm based on transactional confidence applied to the example data set.

Model based on TIC	Filtered rules	Recommendations
1. a 1.00, b 1.00, c 1.00, d 1.00	1. **a** 1.0, **b** 1.00, c 1.00, d 1.00	d: 3.00
2. d 1.00, a 0.50, e 0.50	2. d 1.00, **a** 0.50, e 0.50	c: 2.00
3. d 1.00, e 0.67		e: 0.50
4. a 1.00, b 0.67		

Algorithm 5: Recommendations based on pairwise association rules.

```
function Recommend
    input  : PM, pairwise association rules
             IF, foods selected by a respondent
    returns: RF, list of food recommendations
1   RF ← ∅;
2   P ← ∅, conditional probabilities of foods;
3   W ← ∅, conditional probability weights;
4   OD ← PM[OD], food occurrences ;
5   CD ← PM[CD], food co-occurrences ;
6   foreach input food inf ∈ IF :
7       foreach food f ∈ CD[inf] & f ∉ IF :
8           if f ∉ P & f ∉ W :
9               P[f] ← ∅ ;
10              W[f] ← ∅
11          p ← CD[inf, f] / OD[inf];
12          P[f] ← P[f] + {p};
13          W[f] ← W[f] + {OD[inf]}
14  foreach food f ∈ P :
15      RF[f] ← sum(P[f]) ∗ sum(W[f])
16  return RF
```

ent associations are summed into its single recommendation score $P[f]$.

As demonstrated in Roth et al. (2010) the number of times items are observed together is an important relevance metric. Indeed, if we simply aggregate the probabilities derived from multiple associations, we lose information as to whether a recommended food has ever been observed with all *IF*. For example, given two input items C and D, the aggregation may produce two scores $R_{CD}(A) = 0.5$, where item A appeared with both C and D, and $R_C(B) = 0.5$, where item B appeared only with item C. Therefore A should receive a higher score. Likewise we take into account the frequency of an input food *inf* that matched a retrieved pair. For example, we may have two equal scores, $R_C(A) = 0.5$ and $R_D(B) = 0.5$, where A and B historically appeared only with items C and D respectively; but C appeared 10 times and item D appeared 100 times, which implies that the recommendation produced by C should have a higher score. For these reasons, the algorithm weights aggregated probabilities $P[f]$ by multiplying them by the summed frequency of *inf*.

Recommendations produced by PAR applied to the example transaction data set {*abcd, ade, de, ab*} given items {*ab*} are provided in Table 3.

4. Methodology

We compare the three algorithms for 20000 randomly sampled meals, each containing no fewer than two foods, reported by participants of various ages in the UK between 2014 and 2018. We also randomize the order of foods in each meal. We use k-fold ($k = 10$) cross validation to segment the data set into training and testing sets (Salzberg, 1997). On each step we use nine subsets for training a model, leaving out one subset for testing. The testing procedure is similar to the procedure described in Roth et al. (2010): we sample a few foods from each meal (input foods), leaving the rest (at least one food) to simulate respondents' omitted foods to be guessed by the algorithm. Every trained model makes predictions, starting from an input size of one food and gradually incrementing it to five.

In the course of the evaluation, we plot the precision-recall (PR) curves for every algorithm on every increment. For the purposes of the evaluation, we measure the recall as the percentage of correct predictions out of the total number of foods selected by the respondent, and the precision as the percentage of correct predictions out of the total number of predictions made by the algorithm. We count predictions that were present in the set of foods actually entered by the respondent (excluding input foods) as correct (true positives). We analyze the quality of the top 15 recommendations, which is a slightly larger size than viewed by most users (Burges et al., 2005; Van Deursen & Van Dijk, 2009). As the measure of algorithm ranking quality for every size of input foods we calculate the mean value of Normalized Discounted Cumulative Gain (nDCG) at rank 15 (Burges et al., 2005) as $nDCG_{15} = DCG_{15}/IDCG_{15}$. Discounted cumulative gain is measured as $DCG_{15} = \sum_{i=1}^{15}(2^{r(i)} - 1)/\log(i+1)$, where $r(i)$ is the relevance score of the ith food. As the relevance score, we use 0 for a wrong prediction and 1 for a correct prediction. Thus, the Ideal Discounted Cumulative Gain (IDCG) in our case is always 1, which is a single correct prediction as the first result. We then select the implementation that demonstrates the highest performance and apply it to the task of recommending foods omitted by respondents with a lower level of specificity, and for ranking search results returned in response to their text queries.

For the implementation of AR we use the FP-growth algorithm (frequent patterns algorithm) (Li & Deng, 2007). FP-growth is an efficient and scalable association rules mining algorithm that is based on building frequent-pattern tree structure. In contrast to Apriori-like algorithms that serve the same purpose, the FP-growth compresses a large database into a much smaller data structure avoiding costly repeated database scans and generation of a large number of candidate sets. We use a parallel version of FP-growth implemented in the Apache Spark framework (Li, Wang, Zhang, Zhang, & Chang, 2008; Meng et al., 2016). As a parameter, this implementation accepts the minimum support for an item set to be identified as frequent and the minimum confidence for the generated association rules. To gather as many association rules as possible we set both the minimum support and the minimum confidence to the lowest value (3×10^4) that allows the completion of the mining process of our data set on our machine within a time limit of 5 minutes. The evaluation is conducted on Mac Pro (2.9 GHz Intel Core i5, 16 GB).

5. Results

5.1. General performance

As can be observed from the PR curves (Figs. 1 and 2), PAR produces the largest area under the curve, which increases with the

Table 3
Recommender algorithm based on pairwise association rules applied to the example data set.

Model based on PAR	Filtered rules	Recommendations
1. a 3.0 ⇒ b 2.0, d 2.0, c 1.0, e 1.0 2. b 2.0 ⇒ a 2.0, c 1.0, d 1.0 3. c 1.0 ⇒ a 1.0, b 1.0, d 1.0 4. d 3.0 ⇒ a 2.0, e 2.0, b 1.0, c 1.0 5. e 2.0 ⇒ d 2.0, a 1.0	1. **a** 3.0 ⇒ d 2.0, c 1.0, e 1.0 2. **b** 2.0 ⇒ c 1.0, d 1.0	d: 5.8 c: 4.2 e: 1

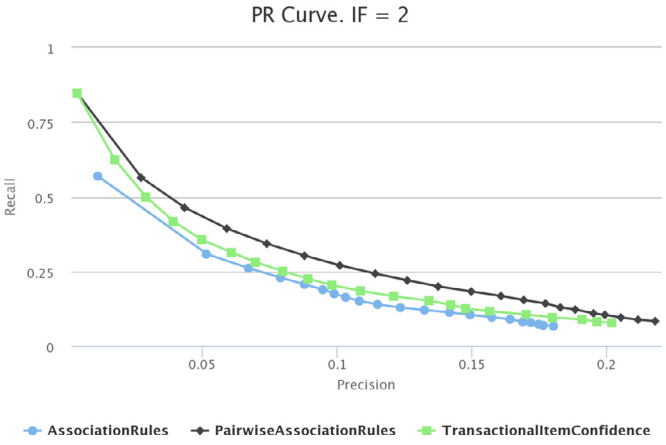

Fig. 1. Precision-Recall curves for an input size of 2 foods.

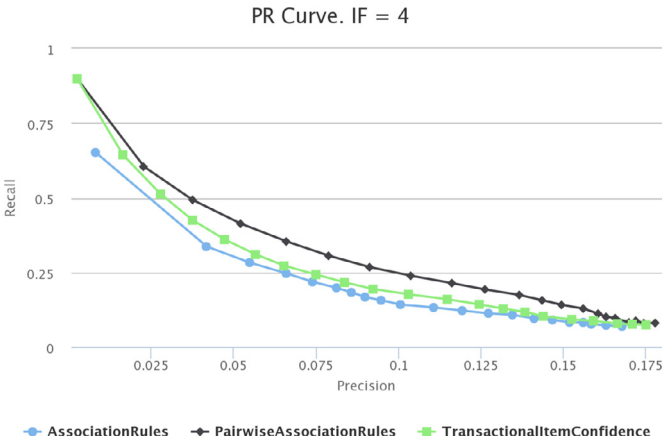

Fig. 2. Precision-Recall curves for an input size of 4 foods.

Table 4
Mean training and recommendation times in milliseconds.

Model	Training	Mean recommendation
AR	3905.1	39.5
PAR	6904.9	2.5
TIC	93710.2	32.0

size of input foods. PAR also demonstrates higher nDCG than TIC and AR for all input sizes (Fig. 3). PAR is the second fastest algorithm to produce a model (after AR) but the fastest to produce a single set of recommendations (Table 4). Based on this comparison, PAR is selected to be used for the implementation of the associated foods recommender algorithm. At the same time, these results demonstrate that the quality of predictions produced by PAR is still relatively low. In the following experiments we aim to improve the performance of the algorithm by exploiting the context of the task it is used for.

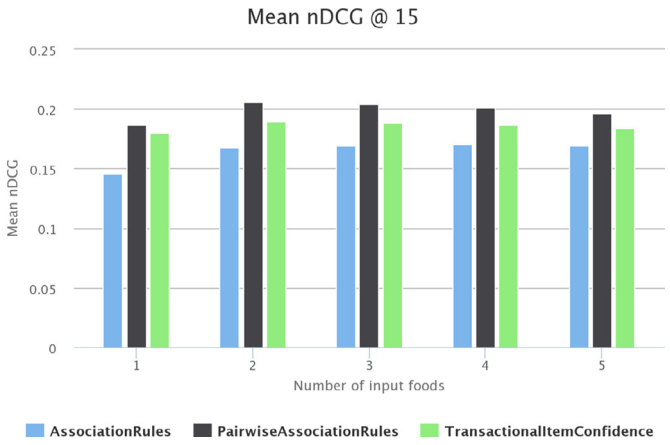

Fig. 3. The ratio of mean nDCG for the top 15 results to the number of input foods.

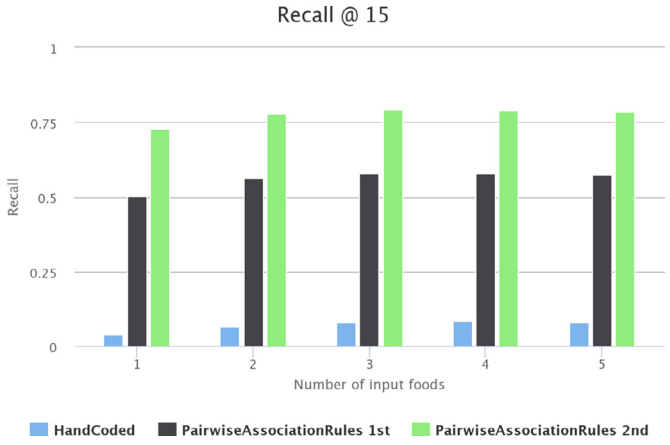

Fig. 4. The ratio of recall for the 15 results to the number of input foods for pairwise association rules with the first and the second levels of specificities and manually entered associated food prompts.

5.2. Associated food questions

To compare the efficacy of recommendations produced by the recommender algorithm to the existing hand-coded associated food questions we go through the same evaluation procedure as above, except that on the recommend step a trained model returns food categories instead of exact foods. In this case, true positives are considered to be foods selected by the respondent (excluding input foods) that belong to one of the food categories predicted by the recommender algorithm. The taxonomy of foods implemented in Intake24 allows control of the specificity of the returned categories. So, we demonstrate the performance of the algorithm in returning the direct parent category of a food (first level, e.g. *Flake cereals* is the parent category for *Choco flakes*) and a more generic category (second level, e.g. *Breakfast cereals*) that is a parent of the category with the first-level specificity (Fig. 4). Since the existing

Table 5
Omitted foods captured with pairwise association rules but not with manually entered associated food prompts.

Input foods	First-level specificity	Second-level specificity
Chicken breast; Fanta; Instant potato	Gravy	Sauces, condiments, gravy and stock
Bananas; Fruit and yoghurt smoothie; Semi skimmed milk	Sugar	Sugar, jams, marmalades, spreads and pates
Blackcurrant squash (juice), e.g ribena; Heinz beans and sausages	Brown bread toasted	Brown, wholemeal and 50:50 bread
Porridge, made with skimmed milk; Tea; White sugar	Butter	Butter/margarine/oils
Tuna mayo sandwich; Volvic mineral water, still or fizzy	Chocolate covered biscuits	Sweet biscuits
Bread sticks; Coffee	Dips	Pickles, olives, dips and dressings
Cheese and tomato pizza (includes homemade); Raspberries	Ice cream	Ice cream & ice lollies
Cheese sandwich; Tea	Crisps and snacks	Crisps, snacks and nuts
Green Olives; Water	Wine	Alcohol
Bottled mineral water; Chicken breast fillet; Chips, fried; Hot sauce	Fizzy drinks	Drinks
Still energy drink, eg Lucozade Hydroactive, Gatorade, Powerade; Tuna in brine, tinned; White bread sliced	Mayonnaise	Sauces/condiments/gravy/stock

associated food questions do not store any relevance scores plotting their PR curves or assessing their nDCG is impossible. For that reason we compare the recall of the top 15 recommendations produced by the algorithm to the recall of all hand-coded associated foods rules extracted for given input foods.

In the simulation of respondents omitting foods hand-coded association food rules recognize 8.3% of omitted foods at most, whereas the recommender *algorithm's* peak recall is at 58.0% and 79.1% for the first and the second levels of specificity respectively.

Table 5 includes examples of commonly forgotten foods established in the validation of Intake24 (Bradley et al., 2016) but correctly predicted by the recommender algorithm with two levels of specificity. At the time of writing this paper, none of these associations were covered by hand-coded associated foods rules in Intake24. In addition to that, controlling the specificity of the returned recommendations allows us to address the cold-start problem, so that new foods that have not been reported by any respondents can still be captured by their categories. However, the names of some food categories predicted with the second-level specificity could be perceived as too generic (e.g. "Pickles, olives, dips and dressings") and may require being assigned names that would be easier to understand by respondents when displayed in associated food prompts.

5.3. Search ranking

In response to a respondent's text query, the existing Intake24 search algorithm ranks foods based on two types of scores. The first is the matching cost of the known food description against the query. The matching cost is based on several metrics, including the edit distance between matched words (the approximate string matching is performed using Levenshtein automata Burges et al., 2005); phonetic similarity of words (using a pluggable phonetic encoding algorithm that depends on the localization language, e.g. Soundex or Metaphone for English Elmagarmid, Ipeirotis, & Verykios, 2007); the relative ordering of words; the number of words not matched; and so forth. The lower the matching cost, the better the food name matches the query. The second score is the likelihood of the food being selected, which is measured by the number of times the food was previously reported. The results are then sorted, first by decreasing food report count (FRC) and then by increasing matching cost.

The evaluation of the associated foods recommender algorithm applied to the task of ranking search results, follows the same evaluation procedure, with some variations. In response to each text query that was recorded into the Intake24 database for each reported food (excluding input foods), we retrieve a list of foods using the existing search algorithm. We count foods selected by a respondent as true positives and the rest of the results as false negatives. We compare the mean nDCG produced by the existing search algorithm and by the new search algorithm, where FRC is replaced with PAR. As we can see from the figure below (Fig. 5), PAR slightly outperforms FRC starting from an input size of two foods, with the gap gradually widening as the number of input foods increases.

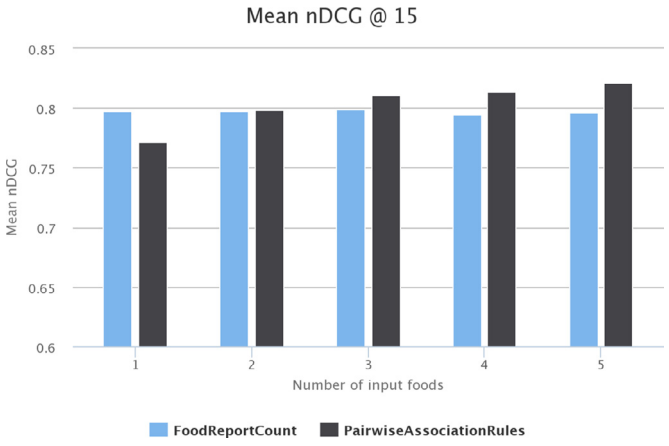

Fig. 5. The ratio of mean nDCG for the top 15 results to the number of input foods for the search results ranked based on pairwise association rules and FRC.

6. Conclusions

We aimed to address one of the key issues in automated dietary assessment, which is unintentional under-reporting. To do so, we developed an associated foods recommender algorithm to remind respondents of omitted foods and improve the ranking quality of search results returned in response to respondents' free-text food name queries. The algorithm, in contrast with collaborative and content-based filtering approaches, is independent of personal user profiles and does not require an extensive history of users' preferences or a multitude of item descriptors. Instead, the algorithm uses transactions performed by respondents from a given population to build a collective model of preferences.

We considered three approaches to the implementation of the recommender algorithm, based on an implicit social graph (Roth et al., 2010), association rules (Agrawal et al., 1993), and analyzing pairwise association rules (DuMouchel & Pregibon, 2001). The evaluation, performed on a large data set of real dietary recalls, has demonstrated that the implementation based on pairwise association rules performs better for the defined task. By controlling the specificity of the produced recommendations within a reasonable level we achieved a recall of 79.1%. That is significantly higher than food associations hand-coded by trained nutritionists, the recall for which reached only 8.3%. Where a respondent filled in at least one food, the recommender algorithm improves the ranking of search results.

The algorithm was evaluated on dietary recalls of respondents from the UK. As a future work we are planning to analyze how dietary specificities of different regions affect the accuracy of the recommender algorithm. Although the evaluation results described in this paper were produced by analyzing food contents of meals reported by respondents in Intake24, the described methods apply to any recommender tasks where selection of items by the target user can be observed (e.g. email recipients or tags recommendations on community platforms).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.eswa.2018.07.077.

References

Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proc. 1993 ACM SIGMOD international conference on management of data, SIGMOD'93: 22* (pp. 207–216). ACM. doi:10.1145/170036.170072.

Atkinson, J., Figueroa, A., & Pérez, C. (2013). A semantically-based lattice approach for assessing patterns in text mining tasks. *Computación y Sistemas, 17*(4).

Bradley, J., Simpson, E., Poliakov, I., Matthews, J. N. S., Olivier, P., Adamson, A. J., & Foster, E. (2016). Comparison of INTAKE24 (an online 24-h dietary recall tool) with interviewer-led 24h recall in 11–24 year-old. *Nutrients, 8*(6), 358. doi:10.3390/nu8060358.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on machine learning* (pp. 89–96). ACM. doi:10.1145/1102351.1102363.

Carrer-Neto, W., Hernández-Alcaraz, M. L., Valencia-García, R., & García-Sánchez, F. (2012). Social knowledge-based recommender system. Application to the movies domain. *Expert Systems with Applications, 39*(12), 10990–11000. doi:10.1016/j.eswa.2012.03.025.

Covington, P., Adams, J., & Sargin, E. (2016). Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems, RecSys '16* (pp. 191–198). ACM. doi:10.1145/2959100.2959190.

DuMouchel, W., & Pregibon, D. (2001). Empirical bayes screening for multi-item associations. In *Proceedings of the 7th ACM SIGKDD international conference on knowledge discovery and data mining, KDD'01* (pp. 67–76). ACM. doi:10.1145/502512.502526.

Elmagarmid, A. K., Ipeirotis, P. G., & Verykios, V. S. (2007). Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering, 19*(1), 1–16. doi:10.1109/TKDE.2007.250581.

Li, C. R. J., & Deng, Z. H. (2007). Mining frequent ordered patterns without candidate generation. In *Proceedings 4th international conference on fuzzy systems and knowledge discovery, FSKD 2007: Vol. 1* (pp. 402–406). IEEE. doi:10.1109/FSKD.2007.402.

Li, H., Wang, Y., Zhang, D., Zhang, M., & Chang, E. (2008). Pfp: Parallel fp-growth for query recommendation. In *Proceedings of the 2008 ACM conference on recommender systems* (pp. 107–114). ACM. doi:10.1145/1454008.1454027.

Lika, B., Kolomvatsos, K., & Hadjiefthymiades, S. (2014). Facing the cold start problem in recommender systems. *Expert Systems with Applications, 41*(4 PART 2), 2065–2073. doi:10.1016/j.eswa.2013.09.005.

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing, 7*(1), 76–80. doi:10.1109/MIC.2003.1167344.

Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., et al. (2016). Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research, 17*(1), 1235–1241. doi:10.1145/2882903.2912565.

Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In *The adaptive web: Vol. 4321* (pp. 325–341). Springer. doi:10.1007/978-3-540-72079-9_10.

Popescul, A., Pennock, D. M., & Lawrence, S. (2001). Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Proceedings of 17th conference on uncertainty in artificial intelligence* (pp. 437–444). Morgan Kaufmann Publishers Inc.

Raeder, T., & Chawla, N. V. (2011). Market basket analysis with networks. *Social Network Analysis and Mining, 1*(2), 97–113. doi:10.1007/s13278-010-0003-7.

Roth, M., Ben-David, A., Deutscher, D., Flysher, G., Horn, I., Leichtberg, A., et al. (2010). Suggesting friends using the implicit social graph. In *Proceedings of 16th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 233–242). ACM. doi:10.1145/1835804.1835836.

Rudin, C., Letham, B., Salleb-Aouissi, A., Kogan, E., & Madigan, D. (2011). Sequential event prediction with association rules. In *Proceedings of 24th annunal conference on learning theory* (pp. 615–634).

Ruiz, P. P., Foguem, B. K., & Grabot, B. (2014). Generating knowledge in maintenance from experience feedback. *Knowledge-Based Systems, 68*, 4–20.

Salzberg, S. L. (1997). On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery, 1*(3), 317–328.

Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '02* (pp. 253–260). ACM. doi:10.1145/564376.564421.

Sene, A., Kamsu-Foguem, B., & Rumeau, P. (2018). Discovering frequent patterns for in-flight incidents. *Cognitive Systems Research, 49*, 97–113.

Shaw, G., Xu, Y., & Geva, S. (2010). Using association rules to solve the cold-start problem in recommender systems. In *Pacific-Asia conference on knowledge discovery and data mining: 6118* (pp. 340–347). Springer. doi:10.1007/978-3-642-13657-3_37.

Van Deursen, A. J., & Van Dijk, J. A. (2009). Using the Internet: Skill related problems in users' online behavior. *Interacting with Computers, 21*(5–6), 393–402. doi:10.1016/j.intcom.2009.06.005.

Zheng, Z., Kohavi, R., & Mason, L. (2001). Real world performance of association rule algorithms. In *Proceedings of 7th ACM SIGKDD international conference on knowledge discovery and data mining, KDD'01* (pp. 401–406). ACM. doi:10.1145/502512.502572.