Ranking Item Features by Mining Online User-Item Interactions

Sofiane Abbar[†], Habibur Rahman[‡,†], Saravanan Thirumuruganathan[‡,†], Carlos Castillo[†], Gautam Das[‡,†]
[†]*Qatar Computing Research Institute* [‡]*University of Texas at Arlington*
sabbar@qf.org.qa, {habibur.rahman@mavs,saravanan.thirumuruganathan@mavs}.uta.edu,
chato@acm.org, gdas@cse.uta.edu

***Abstract*—We assume a database of items in which each item is described by a set of attributes, some of which could be multi-valued. We refer to each of the distinct attribute values as a feature. We also assume that we have information about the interactions (such as visits or likes) between a set of users and those items. In our paper, we would like to rank the features of an item using user-item interactions. For instance, if the items are movies, features could be actors, directors or genres, and user-item interaction could be user liking the movie. These information could be used to identify the most important actors for each movie. While users are drawn to an item due to a subset of its features, a user-item interaction only provides an expression of user preference over the entire item, and not its component features. We design algorithms to rank the features of an item depending on whether interaction information is available at aggregated or individual level granularity and extend them to rank composite features (set of features). Our algorithms are based on constrained least squares, network flow and non-trivial adaptations to non-negative matrix factorization. We evaluate our algorithms using both real-world and synthetic datasets.**

I. INTRODUCTION

The fundamental maxim of any successful business is "Know Your Customer". In other words, knowing (a) *what* are the items that users like? and (b) *why* do they like them? Previously such understanding involved tedious and time-consuming methods such as customer surveys, focus groups or site visits. However, the advent of web has now enabled businesses to easily quantify *what* items users like by measuring online interactions between users and the items. Such interactions might involve users visiting the item webpages or rating them. Recently, it has become easy for businesses to put Facebook "Like" buttons[1] or "+1" buttons[2] from Google over their item webpages which can be clicked by users to demonstrate they like an item.

While the various user-item interactions such as visits, likes, +1s and ratings provide a rich window into *what* users like, answering the second question of *why* a user likes the items is much trickier. If each user had provided elaborate comments or reviews detailing why she liked the item, then identifying the most important features can be solved by extracting semantic information using text mining and information extraction [10], [11]. However, online users are prone to be laconic - the fraction of users that provide detailed comments is minuscule [17]. Hence, the vast majority of user-item interaction information is in its *most rudimentary form* - where we only know whether users visited or liked an item or not.

We consider a database of *items*, where each item is described by a set of *attributes*, some of which are multi-valued. We refer to each of the distinct attribute values of an item as *features* (or equivalently, an item can be described as a set of features). For instance, in a movie streaming site such as Netflix, the items could correspond to movies and attributes might include actors, director, genres, etc. Notice that actors and genres are multi-valued attributes as each movie could have multiple actors/genres associated with it. For an automotive classified site such as cars.com, the items correspond to cars and attributes include the make, model, transmission type, etc. Multi-valued attributes could include its color, customizations etc. Collectively, the set of attribute values forms the features of a given car.

Users are drawn to an item due to a *small subset of its features* [8] and provide a coarse feedback for the entire item by interacting with it. For Netflix, a simple user-item interaction would involve whether the user watched the movie. While some users could have watched the movie because it starred *Tom Hanks*, others could have watched it because, in addition, it was also directed by *Steven Spielberg*. Similarly, while some users might buy a car due to its *manufacturer*, others might buy it for the *model* and *transmission type*.

While significant previous work exists in the general area of mining user-item interactions, our focus in this paper is the investigation of a novel problem: how to *rank* the features of each item from user-item interactions. In particular, we wish to determine a ranking of each item's features where the ranking reflects the contribution of each feature to the popularity (such as number of visits) of the item among the users. Such analysis will enable the owners of the database determining the reasons for the popularity (or lack thereof) of certain items. For example, the Netflix website can get a sense of the most popular actors for each movie, or even the most popular actors globally. A car seller, can identify the set of features of a car that is popular among buyers.

We focus in this paper on the scenario where we have access to *rudimentary* user-item interaction data (for e.g, we only know whether users visited or liked certain items or not). Such information is available at either (a) *individual level*: when the user is identifiable in each interaction (such as user u visited item i), or (b) *aggregated level*: when the user is not identifiable, and we only know the aggregated number of users who interacted with an item.

[1] http://developers.facebook.com/docs/reference/plugins/like/

[2] https://developers.google.com/+/web/+1button/

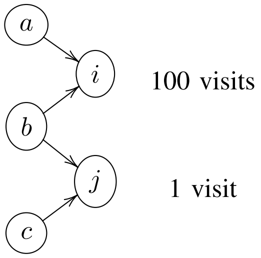


Fig. 1. Database with 3 features and 2 items.

Thus, the principal problem investigated in this paper can be stated as follows:

> FEATURE RANKING (FR) PROBLEM: Given a database where items are described as a set of features, and rudimentary user-item interactions (either at aggregate or individual level), identify the most important features per item (alternatively, a ranked list of features per item).

Challenges. Given a simple user-item interaction data at the level of visits/likes by users, ranking the features of items is a challenging problem. There are no direct cues in the users' feedback that hint about their preferences for certain specific features - e.g., just the fact that a user has seen/liked a movie does not give much insight about the specific actor(s) in the movie that she likes. This indirect expression of preferences makes ordering the features from the aggregate interaction extremely challenging.

Clearly, assuming each feature has equal importance in the popularity of an item is not useful. A more complex approach for identifying the set of most preferred features is what we call the *tag cloud* [16] based approach (as tag clouds in social media interfaces are often generated this way). It associates with each feature a score based on its "popularity" (i.e. cumulative visits of all the items in which the feature is present) and orders them based on the score. However, this method has a number of pitfalls.

Illustrative Example. Suppose we have a database with two items (say movies) i, j which have been visited by 100 and 1 users respectively during a certain time period. Suppose there are three features a,b, and c (say actors) so that a is present in item i, b is present in both items i and j, and c is present in item j. See Figure 1 for a pictorial representation.

Consider the problem of identifying the *global* ranking of features. The *tag cloud* based approach lists the features with a score proportional to their visit frequency. In our example, the score of a (resp. c) will be proportional to the number of visits of i (100) (resp. j (1)), and the score of b is proportional to visits of i and j (101). But if b is actually the most important actor, why item j did not get more interactions? The main problem with this method is the *naïve transfer* of visit count from items to their features, based on the premise that all the features of an item are equally important. Intuitively, in our example there are two possible explanations: (i) the importance of feature a to item i is higher than that of b; (ii) by aggregating the importance of features for all items, feature a has a higher bearing on the potential visits than that of b and c, because otherwise item j would have received more visits. While it is easy to see in our example, generalizing such computations for large data is not trivial and is the focus of this paper.

Our Approach. We propose a probabilistic model that describes user-item interactions in terms of user preference distribution over features and a feature-item transition matrix that determines the probability that an item will be chosen given a feature. Given the observed user-item interaction information at the aggregate (item i was visited by 1000 users) or individual (user u visited item i 10 times) levels, our aim is to estimate the optimal values for user preference distribution and feature-item transition matrix. The individual level is approached as a marginal matrix factorization problem. In contrast to classical non-negative matrix factorization, our problem requires additional constraints such as stochasticity, sparsity and constant values for certain matrix entries requiring non-trivial adaptations. For the aggregate level, we do not have adequate per-user visit information to perform matrix factorization. Instead, we model the problem as estimating the preferences of an "average" user and the corresponding feature-item transition matrix. This is formulated as an optimization problem to which we provide an optimal algorithm based on constrained least squares and a fast approximation based on network-flow.

Summary of contributions.

- For online databases that track user-item interaction information, we introduce and motivate the problem of ranking features of item using only rudimentary user-item interaction information such as user visits.
- Given aggregate user-item interaction information, we propose several algorithms based on *constrained non-negative least squares* and *network-flow*.
- If individual user-item interaction information is available, we design a constrained *marginal* non-negative matrix factorization algorithm where a subset of matrix entries could be constant.
- We present a thorough experimental evaluation of our algorithms using MovieLens/IMDB datasets and study their scalability using large synthetic datasets.

Roadmap. The remainder of the paper is organized as follows. Section II presents our notation and framework. We first discuss the simpler case where each user interacted with the item due to a *single* feature. Sections III and IV describe algorithms for ranking features for the case of aggregate and individual user-item interaction information respectively. In section V, we extend our techniques where user could be interested in a subset of item features. Section VI presents our experimental evaluation over real-world and synthetic datasets. Section VII discusses related work followed by conclusions and future work in section VIII.

II. FEATURE RANKING PROBLEM

In this section, we formalize the data model and describe the two major problem variants that occur in practice when solving the feature ranking problem . For the ease of discussion, we assume that each user interacts with an item because she is interested in a *single* item feature. As an example, user u

watched a movie primarily because it starred actor *Tom Hanks*. We describe extensions to our algorithms where an user is interested in a set of item features in section V.

A. Data Model

Items, Attributes and Features. We assume a database $\mathcal{D}$ of n *items* (tuples). Each item can be represented by a collection of attributes some of which could be multi-valued (such as actors in a movie database). We refer to each of the distinct attribute values of an item as its *features*. In other words, each item can be described as a set of features. The total number of distinct features for the entire database is represented by l.

User-Item Interaction Data. User-item interactions are represented differently at aggregated and individual level. Recall from introduction that we only consider simple interactions such as visits and likes, and not complex ones such as comments. At the aggregate level, such interaction data is represented by an *aggregate interaction vector* v where each component corresponds to the interaction count of all users for a given item. We then normalize the vector to make it stochastic (i.e. with non-negative entries that add up to 1). In the individual case, we have a matrix V where each column provides the interaction count for each item in the database for a specific user. The numerical value V_{ik} reflects the number of interactions between item i and user k. More generally, this represents the relative importance of an item to the user. The matrix V can then be normalized to get stochastic column vectors $V = \{v_1, v_2, \dots v_m\}$ with $v_k \in [0,1]^n$. Given V, we can compute the aggregate interaction vector v as the average of the vectors $v_i \in V$.

B. Terminology

User-Feature Preferences. We denote the number of users of the database by m. We assume a probabilistic user-feature preference model in which a user i expresses her preferences as a probabilistic distribution h_i over features. The preferences of the m users are modelled by an *individual preference matrix* H which contains stochastic column vectors for each user. $H = \{h_1, h_2, \dots h_m\}$ with $h_k \in [0,1]^\ell$. The *aggregate preference vector* of users h is computed as the average of the vectors in H.

Feature-Item Transition Matrix. We assume a column-stochastic matrix $W_{n\times\ell}$ in which rows are items and columns are features. Each cell W_{ij} contains the probability that an average user will visit item i if she is interested in feature j. We assume a known binary matrix $\overline{W}_{n\times\ell}$ which contains the list of all features of each item. We assume that if an item does not contain a feature, then the probability that a user preferring that feature to interact with that item is zero, in other words, $W \leq \overline{W}$. W is not necessarily known, but can be inferred in certain scenarios as described in subsection III-A.

Sparsity assumption. We assume that among all the ℓ features available, each user expresses preference over a relatively small fraction of them: $|h_k| \leq s \ll \ell$. In other words, H is sparse column-wise. As an example, a typical movie goer is most interested in few actors, directors or genres.

Model for User Interactions. Our model for interactions between user and items can be described as follows : the user picks a feature j with probability proportional to $(h_u)_j$. Once she selects a feature, she selects an item i having that feature with selection probability equal to W_{ij}. W_{ij} is the conditional probability that user who has a preference for feature j will visit item i. In short, the model assumes that users first pick some features based on their preference and then selects an item with that feature. The relationships stated above imply that for every user $k = 1, 2, \dots, m$, $Wh_k = v_k$. Additionally, $WH = V$ and $Wh = v$.

Error Measures. Ranking an item's features involves decomposing the interaction information v (resp V) to W and h (resp H). A natural way to verify the accuracy of our methods is via *reconstruction error* – how close the estimated vector/matrix (Wh or WH) is to the observed interactions(v or V). The linear system defined by Wh is overdetermined as the number of equations (one per item) significantly outnumber the number of variables (one per feature). Hence the reconstruction is never exact and only produces an approximate estimate. We denote reconstruction error for our model as $Error(v, Wh)$.

Given two vectors/matrices P and Q, there are a number of commonly used measures to compute the error between an observed and the estimated values :

- L_2 norm : $||Q - P||_2$
- L_1 norm : $||Q - P||_1$
- KL divergence $D(Q||P)$: $\sum_i Q_i \log(Q_i/P_i) - Q_i + P_i$.

C. Problem Variants

In order to rank features of an item, we first need to estimate the feature-item transition matrix W and aggregate preference vector h, or individual preference matrix H. Given a feature transition vector for an item $W_{i\cdot}$ and h, we can compute the relative importance of each feature of the item by performing element-wise multiplication[3] between $W_{i\cdot}$ and h, $X_i = W_{i\cdot} \circ h$. Then the feature ranking is obtained by ordering features based on their value in X_i.

There are two major problem variants depending upon the granularity of user-item interaction information available. In the first variant, the aggregate interaction vector v is available to us and we would like to rank the features of an item i, by estimating item-feature visit vector X_i. We achieve this by computing W and h from v and calculating X_i as $W_{i\cdot} \circ h$. In the second variant, we have in our possession, the individual interaction matrix V and our aim is to compute the vector X_i. We achieve this by decomposing V to its components - feature-item transition matrix W and individual preference matrix H. We then rank the features of an item by computing the aggregate preference vector h from H and then using the equation $W_{i\cdot} \circ h$.

Problem 1 (`FR-AGG`*):* Given a database $\mathcal{D}$ and aggregate interaction vector v, estimate the item-feature visit vector X_i

[3] Given two vectors $A = (a_1, a_2, a_3)$ and $B = (b_1, b_2, b_3)$, we define the element-wise multiplication as the operation that multiplies each component of A with its corresponding one in B, i.e. $A \circ B = (a_1 \times b_1, a_2 \times b_2, a_3 \times b_3)$

(where $X_i = W_{i\cdot} \circ h$) for each item i such that $Error(v, Wh)$ is minimized.

Problem 2 (`FR-INDIV`*):* Given a database $\mathcal{D}$ and individual interaction matrix V, estimate the item-feature visit vector X_i for each item i (where $X_i = W_{i\cdot} \circ h$, h is the average of columns of H) such that $Error(V, WH)$ is minimized.

III. Feature Ranking with Aggregated Interaction Information

In this section, we consider the first scenario where only aggregate interaction information v is available for all items. Recall that the user interaction patterns can be described by equation $Wh = v$, where W is a $(n \times l)$ matrix while h $(l \times 1)$ and v $(n \times 1)$ are vectors. We are interested in item-feature visit vector X_i for each item that can be estimated by the equation $W_{i\cdot} \circ h$. Henceforth, we will focus on computing W and h from v for the rest of the section.

Decomposing a single vector v into a matrix W and another vector h is hard due to the limited information available in v that could possibly be imprecise. This represents a severely under-constrained system where there are $n \times l + l$ unknowns (elements in W and h) that significantly outnumber the number of observed values in v (i.e. n knowns). This admits an unacceptably large space of solutions. Even the sparsity assumption made in §II-C does not contribute to any major simplification of the problem. Hence, we consider two simplified problem variants where we estimate one of the factors (either W or h) via external methods, and solve for the other factor optimally. We first describe an algorithm to estimate aggregate preference vector h given information about W. We then tackle the trickier case of estimating feature-item transition matrix W using h and v. Notice that, for the individual interaction case where $WH = V$ (where W, H and V are also matrices), the problem is solvable via non negative matrix factorization without the need for external methods and is described in Section IV.

A. Feature Ranking via h Estimation

In this section, our aim is to estimate the aggregate preference vector h, which can be viewed as the preferences of an "average" user. In order to estimate h, we require the item-feature transition matrix W and the aggregate interaction vector v. W can be computed using a number of application-specific mechanisms. As an example, if the items were movies and actors were the features, one way to estimate W is to provide different weights to actors depending upon whether had the starring role in the movie. However, if all we have are the boolean feature-to-item matrix $\overline{W}$ (denoting the presence/absence of a feature in an item), we can still approximate W by assuming *uniform* preference to each feature and then making the matrix column-wise stochastic. Given the availability of W and h, this variant of Problem 1 can be formally described as :

Problem 3: Given database $\mathcal{D}$, aggregate interaction vector v and items to features matrix W, determine the aggregate preference vector h that minimizes reconstruction error, $Error(v, Wh)$.

In most databases, the number of items n is significantly higher than the number of features l. In other words, in the system $Wh = v$ the number of equations is significantly higher than the number of unknown variables, resulting in an over-constrained linear system with no solutions. The specific solution will depend on the choice of error function. The generic algorithm to solve Problem 3 is described in Algorithm 1.

Algorithm 1 FR-AGG-W

```
Input: Database $\mathcal{D}$ and aggregate visit vector $v$
$W$ = Estimate feature-item transition matrix
$constraints = \{\ \forall i \in [1, n]\, h_i \geq 0,\ ||h||_1 = 1\ \}$
$h = \underset{h}{\text{argmin}}\ Error(v, Wh)$ subject to $constraints$
Compute $X_i = W_{i\cdot} \circ h\ \forall i \in [1, n]$
return $X = \{X_1, X_2, \ldots, X_n\}$
```

The algorithm tries to solve a constrained optimization problem with non-negativity and stochasticity constraints. Different error functions result in different optimization problems.

A natural error function to use is that of L_2 where we chose the solution that minimizes the reconstruction error $Error(v, Wh)$ defined as $||v - Wh||^2$ (L_2 error). We can notice that this formulation is a variant of the Ordinary Least Squares (OLS) estimator [2] with non-negativity and stochasticity constraints added. There are two major techniques to solve this problem. In the first, the constrained least squares problem is transformed into an alternate formulation that is unconstrained and can then be efficiently solved. Alternatively, the problem can be treated as *generalized singular value decomposition* problem for the compound matrix constructed from W and the constraint matrix.

Complexity. The worst case complexity for computing constrained least squares is $O(n^2 l + n^3)$ [2]. However, a number of efficient iterative algorithms exist that return the solution within a small number of iterations [2].

Example. We compute the aggregate preference vector for the running example described in the Introduction (Figure 1). We constructed W by assuming uniform distribution over preferences. However, even this simple method already provides a more realistic aggregate preference vector for the example of Figure 1. Assuming items i and j are equally important for users interested in their shared feature b, the equations are:

$$\begin{array}{c} \\ i \\ j \end{array}\!\!\begin{array}{c} \begin{array}{ccc} a \quad & b \quad & c \end{array} \\ \begin{pmatrix} 1.0 & 0.5 & 0.0 \\ 0.0 & 0.5 & 0.1 \end{pmatrix} \end{array} \begin{pmatrix} h_a \\ h_b \\ h_c \end{pmatrix} = \begin{pmatrix} \frac{100}{101} \\ \frac{1}{101} \end{pmatrix}$$

Solving this system provides the value for (h_a, h_b, h_c) as $(0.98, 0.015, 0.005)$. Features b and c have a relatively minor importance as otherwise j would have been selected by more users.

The aggregate preference vector h can also be considered as representing the global importance of a feature. Algorithm 1 tries to find the appropriate "score" for each feature based on how well each feature can explain the user-item interaction count. From this perspective, this problem can be considered to be related to the traditional feature selection problem where the features that receive the top score according to some scoring function are selected for model building. However, there exist a number of issues that prevent us from using feature selection techniques. First, the key rationale for feature selection (FS) is to weed out *redundant* features. Further, our reason for computing these scores is to perform feature ranking within each item even when item features are not important globally. There is no easy way to adapt the FS global scores to perform feature ranking that is item dependent. However, our model assumptions allow one to derive item specific feature ranking as described below.

Feature Ranking. Once h is estimated, then the feature ranking of any item can be computed by performing a component wise multiplication between the item's feature transition vector $W_{i\cdot}$ and h. Continuing the running example, the weights of features for item i can be computed as $[1.0, 0.5, 0.0] \circ [0.98, 0.015, 0.005]$ to obtain $[0.98, 0.0075, 0.0]$. Reordering the features based on the weights, we can see that for item i, feature a has higher rank than feature b.

B. Feature Ranking via W Estimation

In this subsection, we describe algorithms to estimate the feature-item transition matrix W. Recall that W_{ij} provides the probability that a user who is interested in feature j will visit item i. Once the matrix W is estimated, it can be used to rank an item's feature using a post processing step. In addition, matrix entries provide an ordering of items that contain a feature which might be of independent interest.

This problem variant is applicable in scenarios where we have access to aggregate preference vector h and aggregate visit information v. h provides the relative importance of the different features for the overall population of users (global ranking of features). This can be obtained from a number of sources, such as domain experts, historical data, polling, indirect evidence such as search volume on a search engine, etc. It must be noted that the ranking must only involve the subset of features that are present in items, i.e. only the features where $\overline{W}_{ij} = 1$. We assume that the presence or absence of a feature in an item is available or can be easily identified. We formally define the variant of Problem 1 below.

Problem 4: Given a database $\mathcal{D}$, aggregate interaction vector v, boolean feature-item presence matrix $\overline{W}$ and a aggregate preference vector h, determine a non-negative item to feature matrix $W \leq \overline{W}$ that minimizes reconstruction error, $Error(v, Wh)$.

We provide two different algorithms for solving Problem 4. First, we consider the case where the L_2 norm is used to compute the reconstruction error, $||v - Wh||^2$ and provide an algorithm FR-AGG-h-LS based on convex quadratic optimization. Even though this algorithm provides an optimal solution, we design an effective and very efficient heuristic by considering the expression $V - Wh$ for computing reconstruction error. The algorithm FR-AGG-h-NF is based on network flow approach. We show experimentally the superior performance of the heuristic with only minor compromise in reconstruction error.

1) Optimal Algorithm using Convex Optimization: The first algorithm we describe is based on an extension of the algorithm for Problem 3. In contrast to estimating the entries of a vector, we need to estimate the nonnegative entries of matrix W.

A natural way of modelling the problem is as a convex quadratic minimization problem with linear constraints. The objective function minimizes the *reconstruction error* defined as $||v - Wh||_2$. The first constraint ensures that only the features that are actually present in the item will be ranked (i.e. features where $\overline{W_{ij}} = 1$). As described previously, such identification can be made by a number of techniques such as content analysis. This constraint also imposes what can be considered as a row-wise sparsity constraint that dramatically reduces the number of unknowns for each item. This is due to the fact that for most items in the database can be described with a small subset of features. The second constraint corresponds to the column-wise stochasticity requirements so that W_{ij} has the interpretation of being the conditional probability that a user interested in feature j will visit item i. Finally, we require that each entry in the matrix W be non-negative.

The optimization problem is a special case of Convex Optimization that can be optimally and efficiently solved by interior point algorithm described in [15]. The algorithms work by adapting Newton's method to *barrier* functions that restrict the solution to the feasible region. A pseudocode for FR-AGG-h-LS is provided in Algorithm 2.

Algorithm 2 FR-AGG-h

```
Input: Database $\mathcal{D}$ and aggregate visit vector $v$
$\overline{W}$ = Estimate feature-item presence matrix
$h$ = Estimate aggregate preference vector
$constraints = \{\ W \leq \overline{W}$ and $\forall j ||W_{\cdot j}||^1 = 1$ and $\forall i, j$
    $W_{ij} \geq 0\ \}$
$W = \underset{W}{\text{argmin}}\ Error(v, Wh)$ subject to $constraints$
Compute $X_i = W_{i\cdot} \circ h\ \forall i \in [1, n]$
return $X = \{X_1, X_2, \ldots, X_n\}$
```

Complexity. Even though the algorithm is known to run in polynomial time, an exact asymptotic runtime is not known. The number of iterations taken is known to be proportional to the square root of the problem size (where problem size is defined as the number of bits of input to the optimization problem). A pessimistic estimate of the worst case running time can be estimated by observing that the problem can be formulated as a constrained least squares. This is done by linearizing the matrix W to a vector w' and converting the h vector to a banded matrix H'. In other words, for each row i, the cells corresponding $i * |l| \ldots ((i + 1) * |l| - 1)$ is set to h and the remaining entries are set to 0. The dimension of w' is $1 \times nl$ and that of H' is $n \times nl$. This results in a runtime that is cubic with respect to the size of database - $O(n^3 l + n^3)$.

Example. Using the running example, we can show the utility of our quadratic minimization based approach. We use the frequencies of features a, b, and c as the input values for h. By solving the optimization using the CVXOPT [1] solver, we get the solution as: $W_{ia} = 1.0$, $W_{ib} = 0.9851$, $W_{jb} = 0.01480$, $W_{jc} = 1.0$.

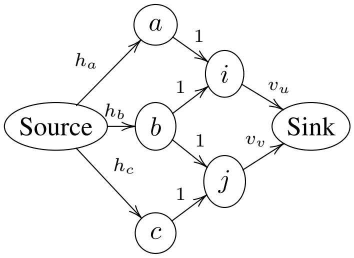


Fig. 2. Augmented graph of the example in Figure 1 for the maximum-flow formulation.

$$\begin{array}{cc} & \begin{array}{ccc} a & b & c \end{array} \\ \begin{array}{c} i \\ j \end{array} & \begin{pmatrix} W_{ia} & W_{ib} & 0.0 \\ 0.0 & W_{jb} & W_{jc} \end{pmatrix} \end{array} \begin{pmatrix} \frac{100}{201} \\ \frac{101}{201} \\ \frac{1}{201} \end{pmatrix} = \begin{pmatrix} \frac{100}{101} \\ \frac{1}{101} \end{pmatrix}$$

2) An Approximate Algorithm using Network Flow: While the solution provided by `FR-AGG-h-LS` is optimal, the time complexity is cubic in the size of the database. For large databases, this is prohibitive and provides a motivation for us to design an efficient heuristic. The primary aim is to design an heuristic that is efficient (has superior running time) and effective (has comparable accuracy). A key challenge in the previous approach was that the optimization, while convex, was quadratic with complex linear constraints. In this approach, we relax the optimization goal from minimizing the reconstruction error to the maximizing the flow of "interest" from features to items.

We consider a graph-based representation of the problem that directly maps to the elements in Figure 1: a bipartite graph where nodes in one partition are features (for instance, actors), nodes in the other partition are items (for instance, items), and an edge exist between an item i and a feature j iff $\overline{W_{ij}} = 1$. Items are associated with their corresponding weights in the aggregate interaction vector v (by capacity-bounded connections to an artificial source node), and features are associated with their corresponding weights in the aggregate preference vector h (by capacity-bounded connections to an artificial sink node. The result is shown in Figure 2.

In comparison with the previous formulation, we switch the objective function from minimizing reconstruction error $||v - Wh||_2$ to that of minimizing $v - Wh$; the reconstruction error becomes one-sided (in other words $v - Wh \geq 0$) and the objective function becomes linear. Notice that this formulation corresponds to a specific class of linear optimization functions equivalent to maximum flow problem. This allows us to tap into its rich literature and use any of the state-of-the-art algorithms for maximum flow problem.

Complexity. We used the Goldberg and Tarjan algorithm with FIFO heuristic [5] that has a complexity $O(nm)$. However, any algorithms for network flow can be utilized as well.

Example. Figure 2 provides the augmented graph corresponding to running example. The output we obtained is $W_{ia} = 1.0$, $W_{ib} = 0.99$, $W_{jb} = 0.01$, $W_{jc} = 1.0$. While the solution is similar to that of previous approach, the algorithm was almost 10 times faster.

Comparison of two approaches. Both `FR-AGG-h-LS` and `FR-AGG-h-NF` tried to compute W given v and h. There are a number of differences between the two approaches. `FR-AGG-h-LS` uses L_2 norm for computing the reconstruction error while `FR-AGG-h-NF` uses L_1 norm. This has the effect that solution provided by `FR-AGG-h-LS` is always optimal. As described previously, the reconstruction error of `FR-AGG-h-NF` is one sided and ($v - Wh \geq 0$ always). This results in both algorithms potentially providing different solutions to the same problem.

We can compute the effectiveness of the solution provided by `FR-AGG-h-NF` by comparing the ratio of its L_2 norm with that of the optimal solution provided by `FR-AGG-h-LS`. Theoretically, this ratio can be as worse as $\sqrt[l]{n}$. However, our empirical experiments show that, in practice, the quality of the solutions provided by both methods are comparable.

Another major factor in the accuracy of the algorithms is the mechanisms used to estimate W (for `FR-AGG-W-LS`) and h (for `FR-AGG-h`). We had previously described few application specific mechanisms to estimate the respective variables. However, the quality of the feature ranking is heavily contingent on the quality of input quantities (h or W). As we will later show in experiments, given an accurate input, our algorithms provide optimal solutions for feature ranking.

IV. Feature Ranking with Individual Interaction Information

In this section we consider the scenario where information about individual users' visits is available. In other words, we possess the matrix V where each column provides the normalized interaction count of each item in the database for a specific user. As the user interaction patterns can be described by equation $WH = V$, we strive to factorize V into the component matrices - a feature-item transition matrix W and an individual preference matrix H. However, recall that our ultimate goal is to compute the item-feature visit vector X_i for each item that can be estimated by the equation $W_{i.} \circ h$, where h is obtained by averaging column-wise vectors of H. We assume that, for each user u, the user-feature preference $H_{.u}$ is sparse, as discussed in section II-C.

Non-Negative Factorization (NMF). Non-negative factorization (NMF) is a powerful technique that was originally proposed as a method for finding matrix factors with parts-of-whole interpretations [13]. Given a nonnegative matrix $R_{n \times m}$ (where all elements have a value greater than or equal to zero) and a positive integer r, NMF aims at finding two nonnegative matrices (factors) $P_{n \times r}$ and $Q_{r \times m}$ such that the matrix product PQ is as close as possible to the original matrix R. This results in an approximation problem where the goal is to minimize an error function that measures the divergence between PQ and R. The two most popular error functions are L_2 norm and Kullback-Leibler divergence.

We will show that our problem of decomposing V into matrices W and H can naturally be expressed as a constrained NMF problem.

Problem 5: Given a database $\mathcal{D}$ and an individual interaction matrix V determine a non-negative feature-item transition matrix $W \leq \overline{W}$ and a sparse individual preference matrix H that minimizes reconstruction error, $Error(V, WH)$ and respects the stochasticity constraints for W and H.

Notice that this problem is different from problems 3 and 4 ($Wh = v$) defined in the previous section where the only available information was an aggregate interaction vector v. The lack of information there made it impossible to solve simultaneously for W and h. In Problem 5, the presence of an individual interaction matrix V along with the sparsity constraints provides enough information allowing to find both factors W and H at once.

Traditionally, NMF algorithms identify both (i) a set of latent features and (ii) their weights. We assume a process wherein we are aware of the features for which users consume an item (e.g., users are drawn to a movie due to some of its actors), and use NMF only to estimate their weights. We believe this use of NMF to identify feature weights but not features themselves is novel in its own right.

Algorithm 3 FR-INDIV-MNMF

```
Input: Database $\mathcal{D}$ and individual interaction matrix $V$
$\overline{W}$ = Estimate feature-item presence matrix
$H_0$ = Initialize a column-wise sparse individual preference
    matrix using setCover (Step 1)
Compute $W_1, H_1$ = M-NMF($\overline{W}, H_0$) (Step 2)
$W, H$ = Impose stochastic constraints (Step 3)
Compute $h = average(H)$
Compute $X_i = W_{i\cdot} \circ h \; \forall i \in [1, n]$ (Step 4)
return $X = \{X_1, X_2, \ldots, X_n\}$
```

Challenges. The classical NMF does not have any constraint other than the non-negativity of matrices W and H, making a direct application of Lee and Sung algorithm [13] inappropriate in our case. In fact, the factors produced by this algorithm may not respect the column stochasticity and sparsity constraints that our factors W and H need to satisfy. Adding general linear constraints to NMF results in a quartic constrained problem that can be solved using the slow "additive" update rules. For instance, Hoyer [9] has proposed a NMF algorithm in which very general sparsity constraints on the factors can be specified. However, perhaps as a consequence of the generality of this algorithm, the proposed complex algorithm uses an "additive" iterative updates based on gradient descent approach (rather than "multiplicative" as in [13]) which significantly slows the convergence. Moreover, there does not appear to be any guarantee of convergence.

Unlike the factors produced by Hoyer algorithm, our factors are subject to a very specific sparsity and stochasticity constraints. The sparsity in H should be column-wise to reflect the fact that a user cannot be interested in all the features of the database. This constraint is captured through $|H_{.u}| \leq s \ll \ell$. The sparsity in W can be seen as set of equality constraints that impose that some entries in W are known and hence should not be update during the factorization process. This constraint is captured through the constraint $W \leq \overline{W}$. Given that some entries are known in the factors makes our problem different from any other NMF algorithm. Hence we refer to our problem as the *Marginal NMF* problem, and develop a factoring algorithm that uses "multiplicative" update rules to speed up the factorization and exhibits provable convergence guarantee. Finally, the stochasticity constraints impose that W rows and H columns sum to one at the end of the factorization.

Our proposed method. We choose Kullback-Leibler divergence $D(V||WH)$ in order to measure the reconstruction error between V and WH. This choice (instead of other measures such as L_2 distance) allows us to design an algorithm that preserves the column stochasticity constraints in the solution. In what follows, we propose a four-step algorithm (Algorithm 3) to solve the problem of ranking item features in the presence of individual interaction matrix V. Our approach has many novel modifications compared to previous algorithms [13], [9] to also handle the specific sparsity and stochastic constraints that the factors W and H need to satisfy. In the first step, we show how to impose sparsity constraints over H using *Set-Cover* techniques. In the second step, we describe the modifications necessary to the NMF algorithm to compute non-negative factors W and H that respect all sparsity constraints. In the third step, we show how to modify W and H to also respect the stochastic constraints. The last step describes how to compute the item-feature visit vector of each item given feature-item transition matrix W and individual preference matrix H.

Step 1: Imposing sparsity constraints over H. Recall from Section II that we impose a (row) sparsity constraint over the factor W by assuming a sparse binary matrix $\overline{W}$ such that $W \leq \overline{W}$. An entry $(\overline{W})_{ij} = 0$ iff item i does not contain feature j. A seemingly similar approach can be used to also impose (column) sparsity constraints over the factor H by defining a sparse binary matrix $\overline{H}$ such that $H \leq \overline{H}$, where an entry $(\overline{H})_{jk} = 0$ if user k has not visited any item that contains feature j. However, this straightforward approach may not generate adequate sparsity constraints, since the union of distinct features of the items that a user has visited may be quite large.

Therefore, we consider a modification of this naive approach, where we consider the items visited by any user k, and derive from this the *minimum* set of distinct features that covers these items. In other words, for each user k we compute column vector $\overline{H}_{.k} \in \{0,1\}^{\ell}$ such that $|\{j : \overline{H}_{jk} > 0\}|$ is minimized subject to

$$\{j : \overline{H}_{jk} > 0\} = \{j : \exists i \; s.t. \; (\overline{W})_{ij} > 0 \wedge V_{ik} > 0\}$$

Computing each column vector $\overline{H}_{.k}$ is equivalent to solving an instance of the *Set-Cover* problem [4]. Although the problem is NP-complete, we use a well-known greedy algorithm that achieves an approximation factor bounded by $\ln(|\{i : (V)_{ik} > 0\}|) + 1$.

The following lemma describes an useful relationship between $\overline{W}$, $\overline{H}$ and V.

Lemma 1: For all i, k, $(\overline{W}\overline{H})_{ik} = 0$ implies $V_{ik} = 0$. In other words, $(\overline{W}\overline{H}) \geq V$

Proof: If the quantity V_{ik} is non-zero, this implies that item i was visited by user k. However, consider the quantity $(\overline{W}\overline{H})_{ij}$. The only way it can be zero is if the set of features in the preference vector of the user k, i.e., $\{r : (\overline{H})_{rk} > 0\}$ is completely disjoint from the set of features that appear in item i, i.e., $\{r : (\overline{W})_{ir} > 0\}$. However, our set-cover based approach of computing $\overline{H}$ prevents this from happening, because if user k has visited item i, then at least one of the features of the item should appear in the user preference vector. ■

Step 2: Iterative algorithm with multiplicative update rules. In the second step, we propose modifications to the algorithm [13] to discover factors W and H such that the reconstruction error $D(V||WH)$ is minimized.

Let us elaborate a bit more on the sparsity constraints. The constraints essentially dictate which entries of W and H should be zeros, while the remaining entries are to be treated as unknown variables which need to be filled with positive values. We thus refer to the former entries as *constants* and to the latter entries as *variables*. Moreover, to make our algorithm (as well as the proof of convergence) simpler to explain, we make a small modification to the sparsity constraints: rather than zero, each constant entry of W and H is assigned a very small constant value of ϵ which remains unchanged throughout the algorithm's execution.[4] Note however, that we do not make any modifications to V, which may contain many zeros.

The algorithm in [13] first initializes W and H to random positive matrices, and then performs the following multiplicative update rules until convergence:

$$H_{jk} \leftarrow H_{jk} \frac{\sum_i \frac{W_{ij} V_{ik}}{\sum_r W_{ir} H_{rk}}}{\sum_i W_{ij}} \tag{1}$$

$$W_{ij} \leftarrow W_{ij} \frac{\sum_k \frac{H_{jk} V_{ik}}{\sum_r W_{ir} H_{rk}}}{\sum_k H_{jk}} \tag{2}$$

We discuss the modifications necessary for applying this algorithm to our problem. First, the initialization step has to be modified. We initialize the constant entries of W and H to a small positive value ϵ (as explained above). We then initialize the remaining variable entries to random positive quantities. Thus, W and H do not contain any zeros.

Next, we note that the updates cannot be applied to all entries of W and H, because then the constant entries will change, thus violating the sparsity constraints. Instead, the update rules are applied *only to the* variable entries of W and H.

The following lemmas and theorem focus on showing that even though the original algorithm in [13] is now restricted to only updating the (variable) parts of W and H, it still has provable convergence characteristics[5].

Lemma 2: After every iteration under the update rules rules (1) and (2), the variable entries of W and H remain non-zero.

Proof: To prove this, we need to show that the multiplicative factor can never become zero. Consider a variable entry H_{jk} and the corresponding update rule (1). The only way in which the multiplicative factor can become zero is if the quantity $\sum_i W_{ij} V_{ik}$ (i.e., the dot product of the two vectors $W_{.j}$ and $V_{.k}$) becomes zero. But the variable entries of W_{ij} represent the set of items that contain feature j, while the variable entries of V_{ik} represent the set of items visited by user k. Since H_{jk} is itself a variable, we know that user k has visited at least one item containing feature j. Thus the two sets overlap, and the dot product of the two vectors $W_{.j}$ and $V_{.k}$ cannot be zero. A similar argument can be used to show that the variable entries in W remain non-zero. ■

Theorem 1: The reconstruction error $D(V||WH)$ is non-increasing under the update rules (1) and (2) when restricted to the variables of W and H.

We do not include the proof due to space constraints. Our proof is obtained by taking into consideration the impact of sparsity constraints in the corresponding theorem in [13], i.e., only the variable entries of W and H are updated. We also show that our update rules are well behaved even if some of the entries V are zero.

Step 3: Imposing stochastic constraints on W and H.

The matrices W and H produced by Step 2 satisfy the sparsity requirements, however, they may not satisfy the column stochastic constraints, which requires that the weights of each column of W and H sum to 1. In this step we describe a procedure for further modifying W and H such that the stochastic constraints are satisfied. We make use of the following theorem by Ho and Dooren [7].

Theorem 2: Let W, H be a stationary point (local minima) for the NMF problem of factorizing V with reconstruction error $D(V||WH)$. Then WH can be further factored into the following form

$$WH = P_{m \times \ell} D_{l \times \ell} Q_{\ell \times n}$$

where P is column stochastic, Q is row stochastic, and D is diagonal non-negative where $\sum_i D_{ii} = \sum_{ij} V_{ij}$. Furthermore, if V is column stochastic, then DQ is column stochastic.

The factorization in the above theorem is accomplished as follows. Define the normalization factors D_W and D_H (both $\ell \times \ell$ diagonal matrices) as the column sum of W and row sum of H, respectively. Then P is obtained by dividing each column of W by its non-zero column sum. Thus $PD_W = W$. Likewise, Q is obtained by dividing each row of H by its non-zero row sum. Thus $D_H Q = H$. If we define $D = D_W D_H$, then clearly $WH = PDQ$.

This theorem immediately suggests the computation that needs to be done in Step 3. We factor the WH returned in Step 2 into PDQ as described above, and since V is known to be column stochastic, we return $W' = P$ and $H' = DQ$. We note that W' and H' satisfy column stochastic *as well as* sparsity constraints.

Step 4: Computing item-feature visit vectors X_i. Once the feature-item transition matrix W and individual preference matrix H are obtained, then the feature ranking of any item can be computed as follows. First, compute the aggregate preference vector h by averaging all column-wise vectors $H_{.j} \in H$, then perform a component wise multiplication

[4] Our algorithm will correctly run even if each constant entry of W and H is assigned the value zero; but the convergence proof is more elaborate.

[5] Strictly speaking, we do not prove convergence; rather we prove that the reconstruction error is non-increasing at each iteration.

between the item's feature transition vector $W_{i.}$ and h, i.e. $X_i = W_{i.} \circ h$.

V. Extensions

In this section, we generalize the user-item interaction model. Previously, our model assumed that user u first picked a *single* feature j based on their individual preference vector h_u and then selected an item i containing j with probability proportional to W_{ij}. However, we now relax that constraint and allow user to pick a *small subset* of features based on their preferences. Intuitively, our prior model dictated that the user watched a movie either because it starred Tom Hanks or it was directed by Steven Spielberg but not for both. We now generalize by allowing such a possibility.

In this paper, we assume that composite features are represented as subset of features, but within each subset the relative weights of features are the same. A more general model of composite features (with varying relative weights of features within each subset) requires decomposing matrix V into three matrices (user-item, composite feature-item, composite feature-weight), which goes beyond NMF and is the focus of future research. Nevertheless, we show in experiments that our algorithms perform reasonably even for our restricted composite features model.

Formally, we allow a tunable parameter p that controls the maximum feature subset size. For eg, a value of $p = 2$ means that users can express their preference over all *individual* and *pairs* of features. We refer to each distinct feature subset as a *composite* feature. Given an item i with features x, y, z and $p = 2$, the composite features are $\{x\}$, $\{y\}$, $\{z\}$, $\{x,y\}$, $\{x,z\}$ and $\{y,z\}$. As mentioned above, we assume that each individual feature within a composite feature has equal weight. i.e. both x and y have equal weight within $\{x, y\}$.

The required changes in our data model is surprisingly simple. While each item is still described by a set of features, the individual preference vector h_u is defined over *composite* features. h_u is a stochastic vector and user picks up a composite feature j with probability proportional to $(h_u)_j$. The feature-item transition matrix W is also now defined over composite features. Each cell W_{ij} provides the probability that an user will consume item i given composite feature j. There is no change in *aggregate* or *individual* interaction information (v or V). The problem variants FR-AGG and FR-INDIV are now defined over composite features where we need to determine *item-composite feature visit vectors* denoted by X_i. All our algorithms are oblivious to the transition between features to composite features.

Feature Ranking with Composite Features. While the concept of composite feature makes our model more realistic, our objective remains ranking the individual features based on their contribution to the item's popularity (such as number of visits). The output of algorithms for problems FR-AGG and FR-INDIV provide item-composite feature visit vectors, X_i. Each component of $(X_i)_j$ can be interpreted as the contribution of composite feature j to the popularity of item i. This can be transformed into a non-stochastic distribution over individual features. Specifically, for each individual feature f and composite feature C, we use the expression $\sum_C (X_i)_C I(f, C)$. Intuitively, we identify all the composite features containing f and aggregate their corresponding component in X_i. $I(f, C)$ denotes a function that returns 1 if $f \in C$ and returns 0 otherwise. Suppose we have $X_{\{a\}} = 0.5, X_{\{b\}} = 0.3, X_{\{a,b\}} = 0.2$ then it can be transformed to individual features as $X^{'}_{\{a\}} = X_{\{a\}} + X_{\{a,b\}} = 0.7$ and $X^{'}_{\{b\}} = 0.5$. Even though the new vector X' is no longer stochastic, it is an estimate of the proportion of user interactions that can be attributed to each individual features.

VI. Experiments

We conducted a comprehensive set of experiments to evaluate the effectiveness and efficiency of various methods for ranking item features. The ranking quality is measured within two scenarios: prediction of the most prominent feature (*precision@1*) and overall ranking of item features (*nDCG@k*). All our predictions are compared to those produced by two different baselines against a predefined ground truth. We show that utilizing the user visit information leads to a better ranking in the case of niche content. In the quantitative evaluation, we measured the execution time (cpu time) of our different methods while varying different parameters. In what follows, we first describe datasets used. Then, we present the ground truth and different baselines and algorithms we implemented. Finally, we report the qualitative and quantitative experiments we conducted and discuss the obtained results.

A. Dataset

We conducted our experiments using the 10M MovieLens dataset[6], which describes the ratings given by a set of users to a set of movies. This dataset is joined with cast data from IMDB[7], indicating which actors acted on each of the movies. The mapping of our problem formulation to this dataset is the following: (i) items are movies, (ii) features are actors in the movies, and (iii) interactions are ratings. We consider all the movies having at least 50 ratings, this yields $464K$ ratings by $5.7K$ users on $1,500$ movies, containing $3,500$ distinct actors.

Synthetic dataset generation. We created different synthetic datasets by varying four parameters: n (# of items), ℓ (# of features), m (# of users), and s (sparsity ratio). The item-feature transition matrix W was generated in such a way that the frequency distribution of features within synthetic collections follows the observed Zipfian distribution of features in movielens. The visit matrix V is created in the same way as W. Finally, the individual preference matrix H is created using several values of column-wise sparsity ratio s.

Implementation details. Our algorithms are implemented in Python 2.7. All the experiments are performed on Linux Ubuntu 12.10 machine, Intel Core i5 2.3 GHz processor, and 12 GB RAM.

B. Algorithms and evaluation metrics

Ground Truth. The IMDB data includes a list of actors sorted by the importance of their roles in each movie, i.e. the first actor is the star of the movie, etc. We use the starring order

[6] http://www.grouplens.org/node/73

[7] http://www.imdb.com

provided by IMDB as our ground truth. We conducted a user study on Mechanical Turk[8] to verify if IMDB based ground truth is representative. For each Human Intelligence Task, we showed a movie and asked users to pick the actor for which they would like to watch the movie. In 92% of cases, users picked the top actor from IMDB listing. To combat worker unreliability, we designed rigorous pre-qualification questions to filter movie savvy workers.

Baselines. We have designed two baselines, *BLtc* and *BLnb* to rank item features. *BLtc* is an extension of the method used to generate tag clouds[16]. It assumes that the importance of a feature is (i) proportional to its popularity in the dataset and (ii) independent from the item in which it appears. For every feature f, we aggregate the number of ratings over items that contain that feature which are normalized to obtain a aggregate preference vector h. The per-item ranking of features is obtained as $X_i = W_{i\cdot} \circ h$, where the weights $W_{i\cdot}$ are assumed to be uniform. The main drawback of *BLtc* is the naive transfer of the number of ratings from items to features.

The second baseline (*BLnb*) overcomes such a problem by adapting a Naive Bayes (NB) algorithm to predict the probability (score) of a feature (f_i) as the fraction of ratings (Y) achieved by items containing it. We first discretized the number of visits Y into five classes (*very popular*,*somewhat popular*,*average*,*not popular*, *niche*) using equi-depth histograms. We assume random variables Y and F_1,F_2,...F_ℓ corresponding to the class of popularity (Y) and feature vector components $f_1, f_2, \ldots f_\ell$. The parameters $P(f_i|Y = y_k)$ for $i \in \{1, 2, \ldots l\}$ and the distribution of $P(Y)$ are estimated using maximum likelihood as follows: $P(f_i|Y = y_k) = \frac{\#I\{F_i=f_i, Y=y_k\}}{\#I\{Y=y_k\}}$ (fraction of items having feature f_i within y_k divided by the cardinality of y_k); $P(Y = y_k) = \frac{\#I\{Y=y_k\}}{|I|}$. The raking of features of an item with y_k ratings is given by the corresponding feature probabilities $P(f_i|Y = y_k)$.

Algorithms. When aggregate interaction information is available we use algorithms `FR-AGG-W-LS`, `FR-AGG-h-LS` and `FR-AGG-h-NF`. The first two use L_2 error function while the latter uses L_1. Algorithms `FR-AGG-h-LS`$\star$ and `FR-AGG-h-NF`$\star$ represent variants of the previous methods where the aggregate preference vector h^* is given by a domain expert. For individual interaction information, we use algorithm `FR-INDIV-MNMF`.

Evaluation metrics. To measure the effectiveness of our algorithms we use *Precision@1* and *nDCG@k*. The former measures the accuracy of predicting the most prominent feature per item whereas the latter measures the quality of item features ranking with respect to the ground truth. *Precision@1* is the fraction of times where the most prominent feature predicted by our algorithm agreed with the ground truth. In our database, this corresponds to identifying the starring actor in a movie. The *nDCG@k* is computed as the ratio of the discounted cumulative gain of the top k most prominent features identified by our methods with that of ground truth. Efficiency of our algorithms is measured in cpu seconds. For each experiment we reported the average of three runs.

C. Qualitative Evaluation

In order to capture the impact of the prolificacy of features on our methods and baselines, we created different subsets corresponding to different prolificacy cut-off(pco) values $\{3, 4, \ldots 12\}$. For instance, a subset with $pco = 3$ contains only those movies whose most prolific actor appears in at most 3 movies. Figure 3(a) illustrates the sizes of the different subsets created.

Figure 3(b) plots the curves of *precision@1* obtained by our algorithms at different pco values. The figure shows that all our methods outperformed both baselines in determining the most prominent feature for niche items ($pco \leq 7$). For instance, in the case of $pco = 3$, `FR-INDIV-MNMF` achieved the best *precision@1* score (0.92) followed by `FR-AGG-h-NF` (0.69) and `FR-AGG-W-LS` (0.58). We notice that from $pco =$ 8 onward, *BLtc* got better predictions than our methods. This is due to the bias introduced by the popularity of some actors. In fact, it is easy to see that an actor who acts in more than 8 movies has a high probability of being the starring actor in the movies he acted in. Further, we can see that our method outperforms *BLnb* in almost all cases.

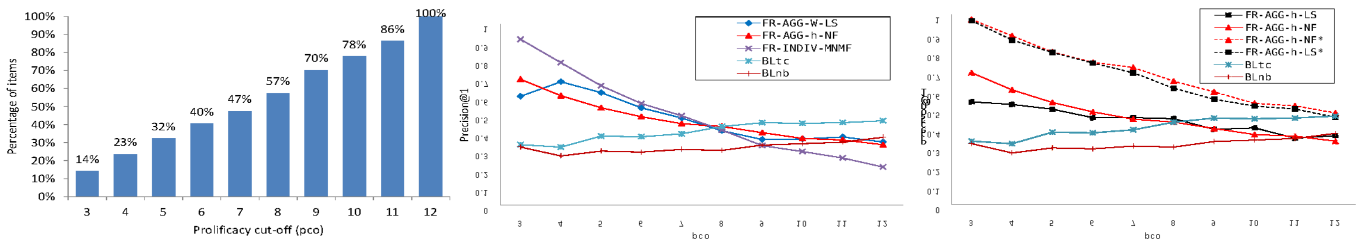
We would like to note that our algorithms are optimized for ranking features in niche items (with low pco) where other methods are inadequate. Items with pco $\geq$ 8 correspond to the most popular movies with disproportionate amount of ratings. The extremely skewed rating distribution distorts the performance of our algorithms. For such cases, tag-cloud based methods provide better results.

Figure 3(c) plots *precision@1* scores obtained by different variants of `FR-AGG-h`. These curves show that whenever a better distribution of features in the aggregate preference vector h^* is known, the accuracy of `FR-AGG-h-LS`$\star$ and `FR-AGG-h-NF`$\star$ is significantly improved. For instance, a peak of $precision@1 = 0.98$ is observed at $pco = 3$. Furthermore, the availability of h^* leads to better predictions even for items with prolific features ($pco \geq 8$). One way to compute h^* is to use the *IMDB* cast information. For a particular movie m, and an actor a acting in it, we compute the score $s_{ma} = 1/i$, where i is the *IMDB* ranking position of a in m. Then, h^* is obtained by summing the scores of every actor a across all movies he acted in.

In general, the quality of all our algorithms can be improved by using domain knowledge if it can be specified as fixed weights in matrix H (user-feature preference) or in matrix W (feature-item dependency). Our algorithms will treat them as constants and estimate the remaining parameters. Handling a more general model of domain knowledge is a topic for future research.

Ranking quality. Figure 4 reports the $nDCG@k$ achieved by our methods within different datasets corresponding to different pco values. Unlike *precision@1*, the $nDCG@k$ captures the quality of ranking the first k features per item. Not surprisingly, our methods achieve a better ranking in the case of niche items (low values of pco). Two observations can be made here. (i) `FR-AGG-h-NF` shows a decreasing trend of $nDCG@k$ scores as k increases. The decreasing trend is amplified in the case of `FR-AGG-h-NF`$\star$. Hence the apt usecase for `FR-AGG-h-NF`$\star$ is to identify the most prominent feature

[8] http://www.mturk.com

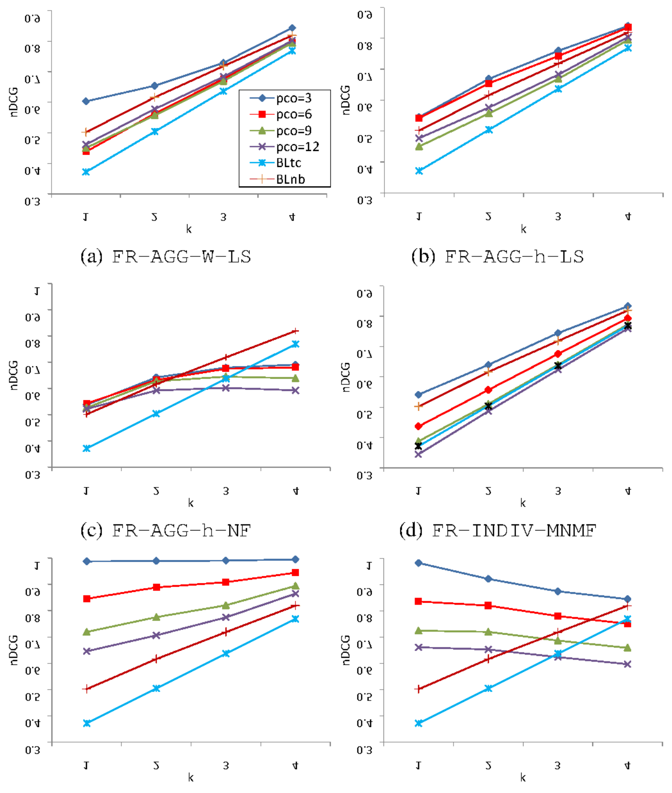


(a) Items distribution function of features prolificacy

(b) Comparative precision@1 curves

(c) precision@1 of different `FR-AGG-h` variants

Fig. 3. (a): size of the dataset at different prolificacy cut-off values. (b) precision@1 for different algorithms. (c) Performance of algorithm variants of `FR-AGG-h` with accurate ground truth h



(a) `FR-AGG-W-LS`

(b) `FR-AGG-h-LS`

(c) `FR-AGG-h-NF`

(d) `FR-INDIV-MNMF`

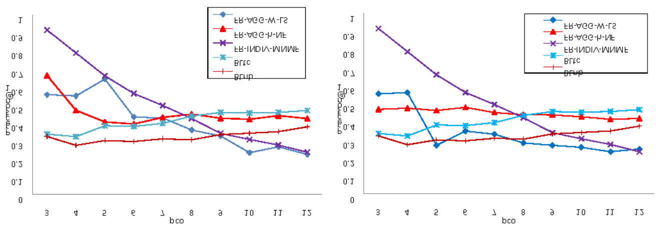
(e) `FR-AGG-h-LS*`

(f) `FR-AGG-h-NF*`

Fig. 4. nDCG@k scores.

(as shown by its excellent performance for *precision@1*). This is caused by the fact that network flow simply tries to maximize the flow without any regard to "spreading" around the credit. (ii) The usage of a good global preference vector (h^*) leads to a significant improvement of *nDCG@k* scores of both `FR-AGG-h-LS*`, and `FR-AGG-h-NF*`.

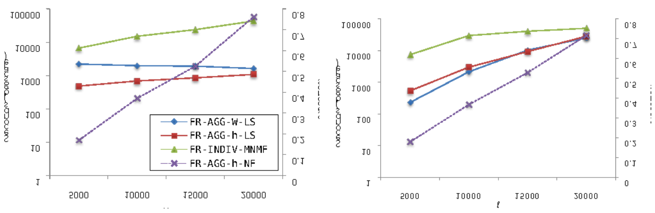
Composite Features. We also evaluated the performance of our algorithms for composite features for the case $p = 2$ and $p = 3$. The results can be seen in Figure 5. We can see that for the movielens dataset, the precision drops slightly for higher value of p. This is to be expected as most users visit a movie for a single prominent reason. Our algorithms still outperform baseline for niche items.



(a) Composite items, p = 2

(b) Composite items, p = 3

Fig. 5. Comparative precision@1 curves.



(a) Varying items

(b) Varying features

Fig. 6. Execution time in cpu seconds.

D. *Efficiency and Scalability*

We analyze the efficiency and scalability of our algorithms by performing a comprehensive set of runs by varying different parameters such as number of items, number of features, and number of users.

Results. In the first experiment, we varied the number of items (n) to take values in $\{5000, 10000, 15000, 20000\}$ while the remaining parameters were fixed as follows: $m = 3000$, $\ell = 5000$, and $s = 0.01$. These numbers result in bipartite graphs with around 1 million edges. Figure 6(A) illustrates the execution time achieved by the different methods. Dashed line curve corresponds to `FR-AGG-h-NF` and should be read on the right-hand side *y-axis*.

Not surprisingly, `FR-AGG-h-NF` is much faster than all other methods in all problem scales. For instance, in the case of $n = 20000$ items, `FR-AGG-h-NF` runs in only 0.8 second. The quartic algorithm(polynomial of degree 4) in `FR-INDIV-MNMF` is the most expensive followed by the quadratic algorithms `FR-AGG-h-LS` and `FR-AGG-W-LS` (with similar curves), while `FR-AGG-h-NF` is almost linear.

In the second experiment we varied the total number of features and results are in Figure 6 (B). The network flow version of `FR-AGG-h` outperforms other algorithms.

TABLE I. EXECUTION TIME OF FR-INDIV-MNMF ON SYNTHETIC DATA VARYING m AND s.

#users (m)	5000	10000	15000	20000
time (sec)	4310.79	9168.48	12105.82	17154.14
sparsity (s)	0.0001	0.001	0.01	0.1
time (sec)	8707.14	8889.79	9707.35	11906.03

In fact, unlike the first experiment where the time achieved by these two methods was almost constant, the runtime of FR-AGG-W-LS and FR-AGG-h-LS increases significantly with ℓ. In other words, increasing ℓ makes the problems more complex by increasing the number of unknowns.

Two more experiments were conducted on FR-INDIV-MNMF by varying the number of users n and the sparsity ratio s. Table I reports the results. For instance, a sparsity value $s = 0.001$ means that a user cannot be interested in more than one per thousand features. The main observation we can make here is that FR-INDIV-MNMF is more sensitive to the number of users than to the sparsity ratio.

VII. Related work

Non negative matrix factorization. Nonnegative Matrix Factorization (NMF) is a powerful tool for data analysis with enhanced interpretability. The seminal paper [13] utilized NMF to identify matrix factors with parts-of-whole interpretations [13] and introduced an iterative algorithm with multiplicative update rules. The cost functions considered were L_2 and Kullback-Leibler divergence. Although many algorithms have been proposed to solve the factorization problem, it seems that none of them satisfies the constraints imposed by our problem. [13] produces factors that may not respect the column stochasticity and sparsity constraints on W and H. While Hoyer [9], proposed a NMF algorithm in which very general sparsity constraints on the factors can be specified, the proposed complex algorithm uses a much slower“additive” iterative steps and does not have convergence guarantees.

Attributes ranking. In recent years there has been some interest in ranking attributes in relational databases. Das et al. [3] orders attributes in order to choose a set of useful attributes that were most influential in the ranking of items. In contrast, our aim is to order the features based on the aggregate or individual interaction count. Miah et al. [14] ranks item attributes so as to maximize the "visibility" of items as defined by the number of top-k queries they match. However, in our problem we are interested in ranking attribute values rather that the attributes themselves.

Feature Ranking. Feature selection [6], [12] is another seemingly related work to ours. The aim of feature selection algorithms is to identify the redundant/irrelevant features that have a low score based on some scoring function that are then discarded. However, our work differs from the extensive work on feature selection because our goal is to rank item features by leveraging user interaction and not to reduce the cost of building models or discard irrelevant features. The scoring function defines a global ranking of features while our aim is to identify item specific feature ranking.

VIII. Conclusions

In this paper, we consider the feature ranking problem that ranks features of an item by only considering user-item interaction information such as visits. We motivated a probabilistic preference model , defined two variants of the problem based on the granularity of the interaction information available and proposed different algorithms (based on constrained convex optimization, network flow approximation, and marginal NMF) to solve these variants. In the future, we would like to investigate a variant where users can choose an item through a weighted combination of features. We would also like to use the sequential user-item interaction information (such as a session) to perform feature ranking.

Acknowledgment

This work was done during the internship of Habibur Rahman and Saravanan Thirumuruganathan at Qatar Computing Research Institute. The work of Habibur Rahman, Saravanan Thirumuruganathan and Gautam Das is partially supported by NSF grants 0812601, 0915834, 1018865, a NHARP grant from the Texas Higher Education Coordinating Board, and grants from Microsoft Research and Nokia Research.

References

[1] M. S. Andersen, J. Dahl, and L. Vandenberghe. Cvxopt: A python package for convex optimization, version 1.1.5. *abel.ee.ucla.edu/cvxopt*, 2012.

[2] Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, 1996.

[3] G. Das, V. Hristidis, N. Kapoor, and S. Sudarshan. Ordering the attributes of query results. SIGMOD '06, pages 395–406.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. 1979.

[5] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. STOC '86, pages 136–146, 1986.

[6] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *JMLR*, 3:1157–1182, Mar. 2003.

[7] N.-D. Ho and P. V. Dooren. Non-negative matrix factorization with fixed row and column sums. *Linear Algebra and its Applications*, 429:1020 – 1025, 2008.

[8] T. Hofmann. Learning what people (don't) want. In *Proceedings of the 12th European Conference on Machine Learning*, EMCL '01, pages 214–225, London, UK, UK, 2001. Springer-Verlag.

[9] P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *JMLR*, 5:1457–1469, Dec. 2004.

[10] M. Hu and B. Liu. Mining and summarizing customer reviews. KDD '04, pages 168–177.

[11] M. Hu and B. Liu. Mining opinion features in customer reviews. AAAI'04, pages 755–760, 2004.

[12] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324, Dec. 1997.

[13] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.

[14] M. Miah, G. Das, V. Hristidis, and H. Mannila. Standing out in a crowd: Selecting attributes for maximum visibility. ICDE '08, pages 356–365, 2008.

[15] Y. Nesterov and A. Nemirovsky. *Interior point polynomial methods in convex programming*, volume 13. SIAM Philadelphia, 1994.

[16] P. Venetis, G. Koutrika, and H. Garcia-Molina. On the selection of tags for tag clouds. In *WSDM*, 2011.

[17] M. Wu. http://lithosphere.lithium.com/t5/science-of-social-blog/The-Economics-of-90-9-1-The-Gini-Coefficient-with-Cross/ba-p/5466, 2010. [Online; accessed 12-Feb-2013].