PNP: Fast Path Ensemble Method for Movie Design

Danai Koutra
University of Michigan
dkoutra@umich.edu

Abhilash Dighe
University of Michigan
adighe@umich.edu

Smriti Bhagat*, Udi Weinsberg*
Facebook
smr,udi@fb.com

Stratis Ioannidis
Northeastern University
ioannidis@ece.neu.edu

Christos Faloutsos
Carnegie Mellon University
christos@cs.cmu.edu

Jean Bolot
Technicolor
jean.bolot@technicolor.com

ABSTRACT

How can we design a product or movie that will attract, for example, the interest of Pennsylvania adolescents or liberal newspaper critics? What should be the genre of that movie and who should be in the cast? In this work, we seek to identify how we can design *new* movies with features tailored to a specific user population. We formulate the movie design as an optimization problem over the inference of user-feature scores and selection of the features that maximize the number of attracted users. Our approach, PNP, is based on a heterogeneous, tripartite graph of users, movies, and features (e.g. actors, directors, genres), where users rate movies and features contribute to movies. We learn the preferences by leveraging user similarities defined through different types of relations, and show that our method outperforms state-of-the-art approaches, including matrix factorization and other heterogeneous graph-based analysis. We evaluate PNP on publicly available real-world data and show that it is highly scalable and effectively provides movie designs oriented towards different groups of users, including men, women, and adolescents.

KEYWORDS

heterogeneous networks; product design; user modeling; user preferences; recommendations; movies

ACM Reference format:
Danai Koutra, Abhilash Dighe, Smriti Bhagat*, Udi Weinsberg, Stratis Ioannidis, Christos Faloutsos, and Jean Bolot. 2017. PNP: Fast Path Ensemble Method for Movie Design. In *Proceedings of KDD '17, August 13-17, 2017, Halifax, NS, Canada, ,* 10 pages.
DOI: 10.1145/3097983.3098076

1 INTRODUCTION

Creating products that satisfy the market is critical to companies as it determines their success and revenue [3]. Online services, such as Amazon, eBay, and Netflix, use data-driven approaches to recommend products to their customers. Recently, however, companies like Netflix have employed their direct knowledge of user viewing habits to make licensing and original programming decisions; this was indeed the case with, e.g. 'House of Cards', one of their most-watched shows [8, 21]. Can we take this decision-making process one step further, and leverage the large amounts of available data (e.g. user ratings or (dis)likes, reviews, product characteristics) *to inform the design of new products and services* that will likely satisfy the target customers?

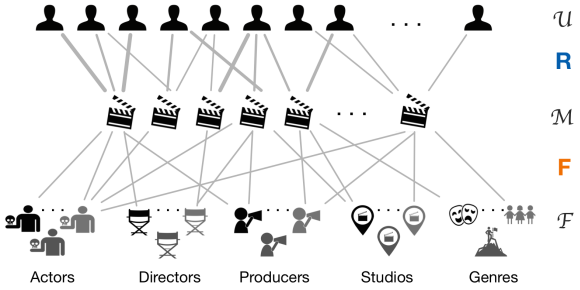


Figure 1: Tripartite graph of the target users $\mathcal{U}$, their rated movies $\mathcal{M}$ and the movie features $\mathcal{F}$.

Market fit for products and services [3], team formation [2, 20], and team replacement [22] are related problems with a similar goal: *designing* a new product, service, or team that will succeed, given a target market or task. The large number of possible choices, combinations, and constraints, as well as the challenge of assessing audience demand, make this a considerably hard problem. Currently, experts use their judgement to estimate solutions to this problem, but this does not scale or allow leveraging massive datasets.

The goal of this paper is to formalize the movie design problem and solve it in a tractable and scalable fashion. Our motivation comes from the movie entertainment industry: We investigate how to design a successful movie that will attract the interest of an arbitrary targeted demographic, subject to budget constraints. Specifically, we tackle the following problem (in short, MD):

Problem 1.[MovieDesign or MD] **Given** (a) a set of users, target users, movies, and features; (b) some user movie ratings; (c) the feature-movie memberships; (d) a cost per feature, and a budget per feature type; **design** a movie which will *attract most of the target users* and is *within budget.*

The MD problem resembles recommender systems [14, 24, 27, 31, 39] and group recommendations [15, 30]. Its main difference is that it does not aim to find the best *existing* movie that the target users are likely to enjoy. Instead, given the user preferences, it

*The work was done while the authors were working at Technicolor Labs.

KDD '17, August 13-17, 2017, Halifax, NS, Canada

DOI: 10.1145/3097983.3098076

determines the features for a *new* movie that is likely to attract the largest number of users.

Solving MD poses several challenges. Clearly, identifying an effective design relies on how movie features affect user preferences. As extensively documented in the recommendation literature [5, 17], and also corroborated by our experiments, "collaborative filtering" approaches and exploiting similarities between users tend to significantly outperform regressing individual user preferences in isolation. It is not immediately clear how to solve MD by leveraging both explicit features and user similarities in a principled, tractable fashion. An additional challenge arises from the fact that the MD problem can be applied to an arbitrary set of users $\mathcal{U}$. Indeed, different strategists may wish to attract varied demographics, and our design should be able to cope with arbitrary, repeated requests. As a result, we would like to efficiently determine movie designs targeting *any* given group.

To address these challenges, we propose a graph-theoretic approach for the MD problem and contribute:

• **Problem Formulation**: We formally pose MD as a problem for designing successful new movies for a target audience. Moreover, we adopt an approach that separates the problem solution into two phases: user-feature preference inference, and model-based design. This separation allows us to infer user preferences and efficiently handle arbitrary "design queries", as defined by targeted user groups.

• **Path-Based Training**: To infer the user-feature scores, we propose a novel model based on predefined walks on a heterogeneous graph consisting of users, movies, and features, which treats "dislikes" in a natural way. This methodology allows us to leverage both features *and* user similarity. Although we focus on the setting of movies, our method is generalizable to any setting where user ratings and product features are available.

• **Model-Based Design**: Having inferred user-feature preferences, we formulate the selection of features that compose a movie as an optimization problem with capacity and other constraints, and establish conditions under which the problem solution is tractable.

• **Experiments:** By using real-world data with ∼5 *million* movie-ratings and 175 000 movie-features, we show that our model of user behavior succeeds in describing user preferences with very high accuracy. In addition, combined with our optimization method, it results in movie design choices that significantly outperform those of our competitors.

In the next section, we introduce our proposed method, PNP.

2 PROPOSED METHOD: PNP

In the MD problem, we assume that the data input consists of a set of users $\mathcal{U}$, a set of movies $\mathcal{M}$, and a set of movie features $\mathcal{F}$. In addition, the feature set $\mathcal{F}$ is partitioned into types (e.g. actors, directors, genres): we denote by $\mathcal{T}$ the set of types, and by $\mathcal{F}_\ell \subset \mathcal{F}$ the set of features of type $\ell \in \mathcal{T}$.

Relations between the above entities are also part of MD's input. These relations consist of: (a) the user-movie ratings, containing tuples of the form (i, j, r_{ij}), where $i \in \mathcal{U}$ and $j \in \mathcal{M}$, and are organized in a $u \times m$ matrix $\mathbf{R}$, with zeros indicating absent ratings; (b) the movie-feature memberships stored in a $m \times f$ binary matrix $\mathbf{F}$, where a non-zero entry (j, k) means that feature k belongs to movie j. Throughout the paper, following observations in the literature [12], we say that user i 'likes' movie j if the rating r_{ij} is larger than i's average rating $\bar{r}_i$ among non-zero entries in $\mathbf{R}$. Table 1 summarizes our notation.

Table 1: Symbols and Definitions. Bold capital: matrices; Lowercase: vectors; Plain font: scalars.

Symbol	Description
G	tripartite input graph
$\mathcal{U}, \mathcal{U}'$	the set of users and target users, resp.
$\mathcal{M}, \mathcal{F}, \mathcal{T}$	the set of movies, features, and types, resp.
u, u'	number of users and target users, resp.
m, f, t	number of movies, features, and types, resp.
$\mathbf{R}$	$u \times m$ matrix of user-movie ratings, with elmnts. r_{ij}
$\mathbf{F}$	$m \times f$ movie-feature membership matrix, with elmnts. f_{ij}
$\mathbf{W}$	inferred $u \times f$ matrix of user-feature preferences
$\mathbf{Q}$	proposed $u \times m$ matrix of user-movie ratings, with elmnts. q_{ij}
τ	threshold in the linear threshold model
$\mathcal{S}$	the set of chosen features
$G(S)$	user conversion function
$\mathbf{x}$	$1 \times f$ binary characteristic vector of $\mathcal{S}$
c_k	cost per feature k (e.g., salary)
B_ℓ	budget per type ℓ (e.g., budget for actors)

The goal of the MD problem (Problem 1) is to design a new movie by selecting its features so that it is within budget and is liked by as many of the people in the target audience as possible. To make the problem tractable, and handle the challenges that we described in the introduction, we reduce it to solving two separate subproblems. We tackle these subproblems in Sections 2.1, and 2.2, respectively.

Problem 2.[Inferring User-Feature Scores] **Given** (a) a set of users $\mathcal{U}$, movies $\mathcal{M}$, and features $\mathcal{F}$, (b) the user movie ratings $\mathbf{R}$, and (c) the movie-feature memberships $\mathbf{F}$; **find** the user-feature preference scores.

The first subproblem amounts to learning user-feature preference scores, capturing the propensity of a given user to like a movie that contains a given feature; the higher the score, the more likely a user is to like this movie. In the second subproblem, we use these scores to formulate the objective of maximizing the number of users liking the movie, among a targeted set.

Problem 3.[Designing the Movie] **Given** (a) a set of target users $\mathcal{U}'$, features $\mathcal{F}$, and types $\mathcal{T}$, (b) the user-feature preferences, and (c) costs c_k per feature k and budgets B_ℓ per feature type ℓ; **select** features for a new movie s.t. is *within budget* and the number of users who will probably like it is maximized.

Next, we discuss our proposed solutions to these two problems.

2.1 Step 1. Inferring user-feature scores

The first constituent problem of MD, Problem 2, can be approached as a classification problem, in which binary 'like' labels of every user are regressed from movie features. Traditional methods (e.g. random forests, matrix factorization) can be used to solve such a problem; as a byproduct, these methods often quantify the effect of each individual feature on a user's decision. However, these methods do not perform well (cf. Section 4) as they either ignore

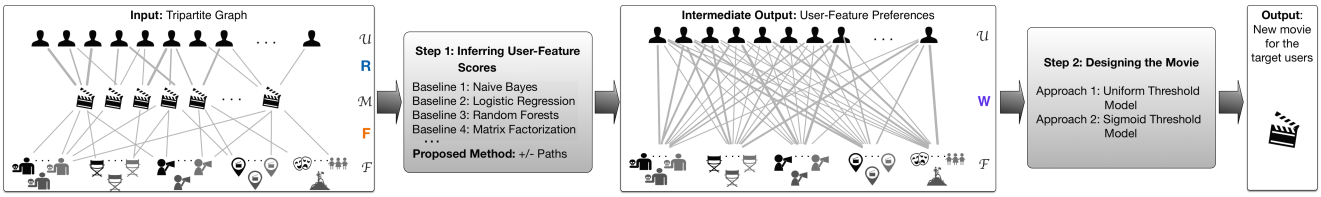
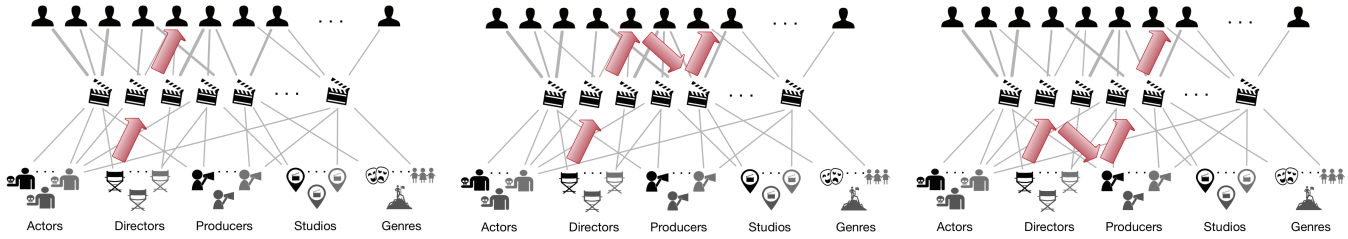


Figure 2: MD: Proposed pipeline for designing a new movie based on user ratings and movie features.



(a) Direct 2-step Path (b) User-based 4-step Path (c) Feature-based 4-step Path

Figure 3: Combination of three types of predefined walks on the heterogeneous tripartite graph. The red arrows show the predefined direction of each walk.

commonalities between users or make use of latent features which cannot be used to identify user-feature preferences.

We propose to solve the MD problem by handling both constituent problems via a graph-theoretic approach, which cleverly leverages *both* explicit features *and* user similarities, and, most importantly, its resulting scores also lead to a tractable movie design optimization for an arbitrary set of target users (Section 2.2). We model the input data as an undirected, heterogeneous, tripartite network where the nodes are users, movies, and movie features, and edges represent the relationships between them. The relationships or edge types are 'rated-by' (for user-movie edges), and 'belong-to' (for feature-movie edges). We give a pictorial overview of our approach in Figure 2.

Specifically, our proposed method, PNP (Positive / Negative Paths), infers the user-feature scores by performing walks of fixed length on the tripartite graph of users, movies and features, thus leveraging information about user-movie ratings as in 'collaborative' approaches, as well as movie 'content' (features). PNP is based on meta-paths, which was first introduced for similarity search in heterogeneous information networks [34]. Informally, a meta-path consists of a sequence of relations between node types. Formally:

Definition. A **meta-path** or **predefined-path** P on a heterogeneous graph G with object types $A_1, A_2, \ldots$ and relations $R_1, R_2, \ldots$:

$$P = A_1 \ldots A_t = A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} A_3 \xrightarrow{R_3} \ldots \xrightarrow{R_{t-1}} A_t$$

defines a composite relation $R = R_1 \circ R_2 \circ \cdots \circ R_{s-1}$ between the source node type A_1 and the target node type A_t, where $\circ$ is the composition operator on relations.

In our setting, the object types include users U, movies M, and features F, and there exist two types of relations: 'rated-by' and 'belongs-to', as well as their inverses.

To capture user-feature preferences, we propose a random-walk-based score restricted over predefined paths starting from F and ending in U. We view the proximity of a user to a feature through such a path as an indication of the user's possible 'preference' towards this feature. For example, a 2-step path via movies captures *direct* user-feature preferences based solely on the movies they have rated. Unlike our approach, the original path similarity [34] is computed as a normalized count of paths between entities. For different types of preferences, we consider three predefined paths:

- **2-step Path**: The path $P = FMU$ (i.e., starting from a feature and ending to a user via a movie) in Figure 3(a) finds the preferences of each user based on her ratings, and, thus, does not exploit a collaborative setting. It computes accurate preferences for the features that appear in movies she has rated, but does not infer preferences for other features.
- **User-based 4-step Path**: The path $P = FMUMU$ in Figure 3(b) computes the user preferences based on the user's input, as well as that of other users who are *similar* to her. The similarity between users is defined via the common movies they have watched; two users who have rated the same set of movies with comparable scores are similar.
- **Feature-based 4-step Path**: The path $P = FMFMU$ in Figure 3(c) finds the user preferences based on the user-movie ratings and *similar* movies to the ones she rated. The similarity between movies is feature-based, i.e., captures commonalities in content.

One way to compute a user-feature score is through the probability that a random walk starting at a given node, restricted to follow only paths of the above three forms, terminates at a given feature. Computing this probability corresponds to matrix multiplications involving the transition matrices induced by each bipartite graph (i.e., user-movies and movie-features). PNP improves upon this approach in two ways, by considering 'positive' and 'negative' random walks, and incorporating edge weights, as described below.

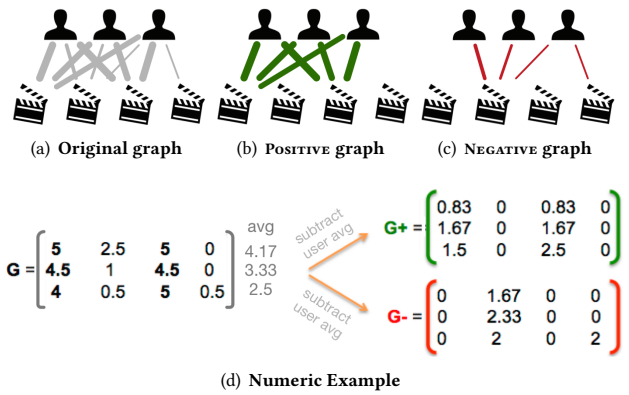


Figure 4: Toy example of Positive and Negative user ratings graphs for our proposed method, PNP.

Positive and Negative Walks. One issue with a random walk approach for preference inference is that the random walk model cannot handle 'dislikes' or 'negative' weights. Constructing the user-movies graph by creating an edge for each rating results in feature scores that only increase in magnitude despite the fact that a rating may be below average, indicating a 'dislike'. However, this is not desired in our MD problem; we want the features that contribute to movies disliked by a user to be penalized by receiving smaller scores than features that contribute to her favorite movies.

To handle this case, we introduce the concept of Positive and Negative walks. Since the movie ratings, which incorporate 'dislike' information, appear in the bipartite user-movie graph, we focus only on this part of the tripartite graph. To obtain the Positive graph from the original bipartite graph, we derive a graph that captures 'like' relations [12], i.e. edges with weight (rating) above the corresponding average user rating. Similarly, we obtain the Negative graph by capturing only 'dislike' relations. The process of generating the Positive and Negative graphs is illustrated in Figure 4.

Edge Weights. To better leverage the user preferences, we assign weights to edges between users and movies on both the Positive and Negative graphs, so that the walker performs a walk over a weighted graph. Since user ratings are often biased [17] (e.g. some users give high scores objectively, even if they did not enjoy the movie), we do not directly use the ratings as edge weights, but we center user i's rating for movie j through $r_{ij} - \bar{r}_i$, where $\bar{r}_i$ is the user's average rating. That is, we adjust the ratings according to each user's average movie rating and refer to it as a 'Centered Rating' (CR). Second, to emphasize big deviations from the average user rating, we use a non-linear reweighing schema:

$$q_{ij} = 2^{\delta * (\mathrm{CR})} = 2^{\delta \cdot |r_{ij} - \bar{r}_i|} \tag{1}$$

which puts more weight on very low and very high ratings. Note that, as ratings are centered around the mean, the notions of low and high are subjective for each user. We denote the resulting $u \times m$ weighted adjacency matrix by $\mathbf{Q}$, and consider random walks over the corresponding weighted graph. We study the effect of the reweighing schema and its non-linearity in Section 4.

PNP Score Computation. Putting all this together, PNP proceeds as follows. It computes the probability that a random walk starting at a user reaches a feature under the following assumptions: the walk is (a) constrained to occur only over paths of the three proposed predefined types, and (b) occurs on either the Positive or the Negative graph, with edges appropriately weighted through Equation (1). Let $\mathbf{w_p}^+$ and $\mathbf{w_p}^-$ be the probability vectors over features, for path type p and user $i \in \mathcal{U}$. Then, we define $\mathbf{w_p} = \mathbf{w_p}^+ - \mathbf{w_p}^-$ as the weighted user-feature preference vector over path type p. It is clear that, as desired, PNP penalizes features in movies that are disliked by a user by subtracting some value from their preference score. We define the final weighted feature vector of PNP for i, which represents i's preference over features in $\mathcal{F}$ as a linear combination of the weighted user-feature vectors over the three predefined path types:

$$\mathbf{w} = \alpha\, \mathbf{w}_{\text{2step}} + \beta\, \mathbf{w}_{\text{4step-usr}} + \gamma\, \mathbf{w}_{\text{4step-feat}} \tag{2}$$

where $\mathbf{w} = (\mathbf{w}^+ - \mathbf{w}^-)$. In matrix form, the feature preferences for all users are:

$$\mathbf{W} = \alpha\, \mathbf{W}_{\text{2step}} + \beta\, \mathbf{W}_{\text{4step-usr}} + \gamma\, \mathbf{W}_{\text{4step-feat}}, \tag{3}$$

where $\alpha,\beta,\gamma \in \mathbb{R}^+$ are the combination parameters satisfying the equation $\alpha+\beta+\gamma=1$, and $\mathbf{W}$ is a $u \times f$ matrix.

2.2 Step 2. Designing the movie

The second constituent problem of MD, Problem 3, tackles the actual selection of features, $\mathcal{S}$, for a new movie so that the number of users who will likely enjoy it is maximized. We propose to formulate Problem 3 as an optimization problem with cardinality constraints and, if available, budget constraints. Following the literature on node-specific threshold models for influence maximization [10, 16], we consider a user i converted with respect to (i.e., 'likes') a movie consisting of features $\mathcal{S}$ if $\sum_{k \in S} w_{ik} > \tau_i$, where w_{ik} is user i's preference score for feature k and τ_i is the user-specific threshold. Treating τ_i as a random variable, we define the user conversion function, $G(\mathcal{S})$, as the expected number of users in the target set $\mathcal{U}'$ that are converted:

$$G(\mathcal{S}) = \sum_{i \in \mathcal{U}'} P[\sum_{k \in \mathcal{S}} w_{ik} > \tau_i] = \sum_{i \in \mathcal{U}'} F[\mathbf{w}_i \mathbf{x}^T] \tag{4}$$

where $\mathbf{x} \in \{0,1\}^f$ is the characteristic row-vector of set $\mathcal{S}$ (i.e., $x_k = 1$ iff $k \in \mathcal{S}$), and $\mathbf{w}_i$ is the row vector that encodes user i's feature preferences. Based on these assumptions, we frame the MD task as a general optimization problem:

$$\begin{aligned} &\text{Maximize} && G(\mathcal{S}) = G(\mathbf{x}) = \textstyle\sum_{i \in \mathcal{U}'} F[\mathbf{w}_i \mathbf{x}^T] && (5)\\ &\text{subject to} && \textstyle\sum_{k \in \mathcal{F}_\ell} b_k\, x_k \le B_\ell, \quad \text{for each } \ell \in \mathcal{T}, && (6)\\ & && \mathbf{x} \in \{0,1\}^f. \end{aligned}$$

In the special case where $b_k = 1$ for all $k \in \mathcal{F}_\ell$, the above problem effectively amounts to maximizing $G(S)$ subject to a *cardinality constraint* $|\mathcal{S} \cap \mathcal{F}_\ell| \le B_\ell$. For example, for type ℓ=actor, the quantity $|\mathcal{S} \cap \mathcal{F}_{actor}|$ is equal to the number of selected actors which is bounded by the maximum number of actors, B_{actor}. Such constraints also make sense as, e.g., no more than one director is

Table 2: Dependencies between movie factors do not seem to correlate with movie success. Analysis of frequent 'itemsets' that have appeared in at least ten movies.

Features	Lasso Regression (5-fold C.V.) MSE (std)	non-0 coeff. (total)	Singletons	Pairs	Triplets	Linear Regression (5-fold C.V.) MSE (std)	non-0 coeff.
singletons	1.0395 (0.0533)	211 (621)	211	n/a	n/a	1.5448 (0.0467)	620
singletons + pairs	1.0303 (0.0328)	207 (969)	161	46	n/a	3.025 (0.3498)	899
only pairs	1.1103 (0.0327)	141 (348)	n/a	141	n/a	4.0232 (17.3369)	100
singletons + pairs + triplets	1.0452 (0.0526)	235 (1104)	179	45	11	2.9254 (0.8767)	951

needed per movie. If the cost c_k per feature is available, then we can set $b_k = c_k$, and, thus, capture budget constraints. We note that the optimization problem has a linear constraint, and the objective depends on the inner product of scores with $\mathbf{x}$; so, it does not model dependencies between features. Next, based on evidence from the real-world data, we explain why we do not model such dependencies.

Why not model feature dependencies? We followed a data-driven approach to evaluate whether dependencies between movie features correlate with movie ratings (e.g. two actors should always play together because then they lead to successful movies, while independently they do not). Using the IMDB dataset described in Section 3, we performed: (1) frequent item mining (a priori algorithm) to find frequent k-itemsets that have appeared in at least ten movies; and then (2) linear and lasso regression with 5-fold cross validation to select the k-itemsets (singletons, pairs or triplets of features) that are most predictive of the movie rating.

Frequent item mining showed that only 1% of the features are frequent (with support $\geq$ 0.5%), half of which contribute to frequent pairs, and 20% contribute to frequent triplets. The results of linear and lasso regression are summarized in Table 2. In general, regression on the dataset with frequent singleton features is better or comparable (in terms of error and model compactness) to regression on datasets that capture feature dependencies. Specifically, the MSE (Mean Squared Error) of linear regression is minimum when the input dataset has only singleton features. The MSE of lasso regression becomes slightly smaller when we consider frequent pairs of features rather than only frequent single features (1.04 vs. 1.03). However, the majority of selected features are singletons, and the number of selected features are almost the same.

Observation 1. The feature dependencies are very few and do not affect the predictability of user-movie ratings significantly.

Uniform Threshold Model. Given its success in modeling influence propagation in networks, we propose to use a linear threshold model where each user i picks her conversion threshold τ_i uniformly at random [16], i.e., we assume no background information about the user conversion thresholds. Since the PNP user-feature preferences computed are in $[-1, 1]$, we set the thresholds to follow the uniform distribution in the same interval, i.e. $\tau_i \sim \mathcal{U}[-1, 1]$. Under these assumptions, the user conversion function (4) becomes

$$f(S) = \frac{1}{2}\mathbf{1}^T\mathbf{W}\mathbf{1} + \frac{|\mathcal{U}'|}{2}$$

Proof. Starting from Equation (4) and applying our assumptions, we obtain:

$$\begin{aligned} f(S) &= \textstyle\sum_{i\in\mathcal{U}'} P[\sum_{k\in\mathcal{S}} w_{ik} > \tau_i] \overset{\tau_i\sim\mathcal{U}[-1,1]}{=} \\ &= \textstyle\sum_{i\in\mathcal{U}'} \frac{1}{2}(\sum_{k\in\mathcal{S}} w_{ik} + 1) = \\ &= \frac{1}{2}\textstyle\sum_{k\in\mathcal{S}}\sum_{i\in\mathcal{U}'} w_{ik} + \frac{|\mathcal{U}'|}{2} \overset{w_k = \sum_{i\in\mathcal{U}'} w_{ik}}{=} \\ &= \frac{1}{2}\textstyle\sum_{k\in\mathcal{S}} w_k + \frac{|\mathcal{U}'|}{2} \end{aligned}$$

Hence, under the uniform threshold model, we want to pick the movie features that maximize $\sum_{k\in\mathcal{S}} w_k$. □

In this case, the problem reduces to knapsack under separable constraints, which, though NP-hard, can be solved with an FPTA scheme [37]. In the unit-cost (i.e., cardinality) case, the problem is solvable in polynomial time: sorting features $k \in \mathcal{F}$ in decreasing order of costs w_k, and picking the top B_ℓ features, is optimal.

Sigmoid Threshold Model. The uniform threshold model has the advantage of resulting in an easy-to-solve optimization problem. To further leverage background information that might be available in the data, data-driven models may be considered for the user-specific thresholds. For instance, if the average rating score per user follows a sigmoid distribution and it is being used as the user-specific threshold, then F in Equation (4) can be replaced with the CDF of the logistic distribution, F_S.

This results in a sigmoidal programming problem [36], which is NP-hard even in its relaxed form with non-integral solution. By formulating the problem as the maximization of $\sum_{i\in\mathcal{U}} F_S[y_i]$, we introduce $|\mathcal{F}|$ additional constraints $y_i = \mathbf{w}_i\mathbf{x}^T$, which can be solved approximately by using the branch-and-bound based method in [36]. The (possibly) fractional solutions can then be converted to integral values solutions by using pipage rounding [1]. However, the approximation error of the solution depends on the number of constraints, and the method may solve many convex optimization problems, which suggests that the method will be impractical for the MD problem with thousands of constraints.

2.3 Complexity of PNP

Naïve approach. Performing the walks over the 2-step and 4-step predefined paths on the positive and negative graphs is computationally expensive and is dominated by sparse matrix and sparse-dense matrix multiplications (e.g., $\tilde{\mathbf{Q}}\tilde{\mathbf{F}}$ for $P = FMU$, or $\tilde{\mathbf{Q}}\tilde{\mathbf{Q}}^T\tilde{\mathbf{Q}}\tilde{\mathbf{F}}$ for $P = FMUMU$, where $\tilde{\mathbf{Q}}, \tilde{\mathbf{Q}^T}, \tilde{\mathbf{F}}$ are $u \times m$, $m \times u$, and $m \times f$ transition matrices). If we assume naïve matrix multiplications, the complexity of performing the walks on the 2-step, user-based 4-step, and feature-based 4-step paths would be: $\Theta(fmu)$, $\Theta(fmu + 2u^2m)$ and $\Theta(3fmu)$. We note that these computations can be trivially

parallelized, so the complexity corresponds to the maximum of the three. Moreover, the output is a dense $u \times f$ matrix $\mathbf{W}$ that captures the user-feature preferences. Although the bulk of the computation can happen offline, we can support quick query processing by just summing up the rows in matrix $\mathbf{W}$ that correspond to the target users $\mathcal{U}'$ and performing linear feature selection in $O(u'f)$ after sorting the scores in decreasing order in $O(f \log f)$. Next, we show that we can significantly speed up the computation and reduce the storage requirements.

Fast approach. The first step of inferring all the user-feature scores results in a dense $u \times f$ matrix $\mathbf{W}$. In the proposed second step of designing a movie, we observe that the uniform threshold model simplifies the problem mathematically, and requires *only the sum* of the target users' feature preferences (i.e., a $1 \times f$ vector that has the sum of *all* the inferred preferences for all target users $\mathcal{U}'$). This observation helps us to significantly speed up the computations and reduce the storage requirements. By using a $u \times 1$ indicator vector $\mathbf{x}$ for the users in $\mathcal{U}'$, all the walks can be re-designed (backwards) as fast sparse matrix-vector multiplications. For example, for the predefined path $P = FMUMU$, instead of computing $\tilde{\mathbf{Q}}\tilde{\mathbf{Q}}^T\tilde{\mathbf{Q}}\tilde{\mathbf{F}}$ in the first step and the aggregate score for the target users in the second step (as in the naïve approach), we can directly compute $\tilde{\mathbf{F}}^T(\tilde{\mathbf{Q}}^T(\tilde{\mathbf{Q}}(\tilde{\mathbf{Q}}^T\mathbf{x})))$ in $\Theta((3\ \text{nnz}(\tilde{\mathbf{Q}}) + \text{nnz}(\tilde{\mathbf{F}}))u')$, where nnz() is the support size (i.e., number of non-zeros) of the corresponding matrix. The intermediate and final computations generate and store vectors of variable lengths. Similar computations can be derived for $P = FMU$ and $P = FMFMU$. The feature selection can be done in $O(f \log f + f)$. As we show in Section 4.1.3, the fast approach is very scalable and up to 100× faster than the naïve approach.

3 DATA

We compiled our data from two different and publicly available sources: (i) the Flixster website [40], a social movie site allowing users to watch and rate movies, from which we obtained user-movie ratings, and (ii) the MGO website [35] which has information about movie features. The Flixster and MGO datasets consist of 66 726 and 83 017 movies, respectively. In addition to the user ratings, the Flixster dataset provides some user demographics, including their gender, age, and location (as free text).

To merge the two datasets, first we dropped the movie remakes. In order to be able to infer the user-feature or user-movie preferences (Problem 1) using 5-fold cross validation, we iteratively filtered the data so that: every movie has been rated by at least 20 users and has at least 2 features; every user has rated at least 20 movies; every feature has appeared in at least two movies. The resulting user-movie-feature tripartite graph has 66 407 nodes and 4 567 253 edges (Table 3). The movie features include: 25 721 actors, 1 322 directors, 4 305 producers, 654 studios, and 27 genres. The user ratings are between 1 and 5, with 0.5 increments.

For our dependency analysis we used an IMDB dataset consisting of 1 893 movies, their features (947 directors, 5 231 producers, 1 209 studios, 51 500 actors, 27 genres, and 4 seasons), and average movie ratings. IMDB is based on the ratings of millions of users, allowing us to observe more global patterns and feature dependencies.

Table 3: Movie data stats (based on Flixster and MGO).

Movies	5,881
Users	28,482
Features	32,029
User-Movie Ratings	4,435,359
Movie-Feature Memb.	131,894

4 EXPERIMENTS

In this section, we give the experimental analysis of PNP, and evaluate its performance for designing new movies for specific audiences. We seek to answer four main questions:

- **Q1.** How does PNP compare to baseline approaches?
- **Q2.** Is PNP robust with respect to its parameters?
- **Q3.** Is PNP scalable?
- **Q4.** How successful are the movies we design?

The first three questions are related to the inference of user-feature-score preferences (Problem 2), and the last question refers to the optimization problem (Problem 3).

4.1 Analysis of Step 1: Inferring feature scores

4.1.1 Q1: PNP vs. baseline methods. To compare the predictive power of PNP against baseline methods, we cast Problem 2 as a binary classification problem where the goal is to predict whether a given user will like or dislike a new movie. More formally, recall that we consider two classes [12]: User i likes (dislikes) movie j if she rated it higher (lower) than her average user score, i.e., C_{ij}=1 if $r_{ij} \geq \bar{r}_i$ (C_{ij}=0 if $r_{ij} < \bar{r}_i$). Each observation corresponds to a user-movie rating, where the independent variables are the movie features (binary), and the dependent variable is the user's preference (like or dislike). We note that the mapping of the user-ratings to binary values results in 55% 'liked' movies, and 45% 'disliked' movies. In all the cases that we describe below, we learn a per-user classifier on their movie ratings, and test it on new movies. To predict the user's preference for a new movie using PNP, we multiply the user-based feature-preference vector $\mathbf{w}_i$ with the new binary movie vector, and compute her movie score.

This problem can be solved by several traditional methods, among which we consider:

Baseline 1.1: Naive Bayes. NB can predict whether or not a user will like a new movie, i.e., it cannot infer the user-feature preferences (Problem 2). NB makes the assumption that the features act independently. To avoid the zero-probabilities issue, we use the Laplace correction.

Baseline 1.2: Logistic Regression. LR is a commonly-used generalized linear model for prediction. Similarly to NB, LR also assumes feature independence. LR also generates a weight vector of the features' contribution in the prediction. Since the number of features is greater than the number of samples (Table 3), we used L1 regularization to ensure sparsity of the produced weight vector. To find the best value of the regularizer parameter, we performed a cross-validated grid-search over the values {0.5, 1, 5, 10}.

Baseline 1.3: Random Forests. RF is an ensemble method for classification and regression, which is robust to the inclusion of irrelevant features and overfitting. For each user, the RF classifier

generates N random subsets on the training data in terms of features, and constructs a separate decision tree for each subset. To predict a new user-movie preference, the movie's feature vector (new sample) runs through all the generated user-specific decision trees, and the mode of the classes is assigned as the predicted class. A drawback of this approach is its runtime, since it requires generating many decision trees per user. In our experiments, for the maximum depth of the trees, we performed grid-search over the values {25, 100, 500, 1000}, and generated 10 random subsets of the data. Overall, RF was very slow despite the small number of grid-search values.

Baseline 1.4: Matrix Factorization. MF [5, 17] with user and item biases predicts the rating of user i for item j:

$$\hat{r}_{ij} = \langle u_i, v_j \rangle + \mu + b_i + b_j$$

where u_i and v_j are d-length vectors (for some small d) that capture the user and item latent vectors, respectively, μ is the global average rating, b_i is the user bias, and b_j is the item bias. The main advantage over the other approaches is that we learn a model for each user using a small dense matrix, where the number of rated movies is commonly larger than the number of features. However, MF operates on a latent feature space, thus using it to design a new movie is not straightforward. We performed MF over the ratings matrix $\mathbf{R}$ to compute user and feature vectors with a dimension of $d = 10$. These were computed using 20 iterations of stochastic gradient descent, and ℓ_2 regularization parameters selected through cross validation. The predicted ratings of user/movie pairs were used in the AUC computations.

Baseline 1.5: Content-based Matrix Factorization. We further extended basic MF by incorporating content-dependent biases: beyond user and movie biases, we include additional bias in our model, one for every item feature. These can be thought of as explicit, rather than latent, features of a movie. We again compute parameters for ℓ_2 regularization penalties through cross validation.

Baseline 1.6: Heterogeneous Entity Recommendation. The method in [38] finds the top-k movies to recommend to a user by performing non-negative matrix factorization on a set of user-movie preference diffusion matrices learned via meta-path similarity [34] and user clustering. We note that this approach cannot be used for *feature* preference inference since the model applies only for entities that are directly linked to and rated by users in the heterogeneous graph (e.g. movies). Thus, in the binary classification task, we use the inferred user-*movie* scores which are directly learned over the metapaths that correspond to our proposed method's predefined paths. We set $k = 20$ for the low-rank approximations and pick the method's parameters through cross validation.

Results. To evaluate the methods, we perform 5-fold cross validation, compute the AUC (Area Under the Curve) per user, and report its average and standard deviation over all users in Table 4.

Observation 2. PNP outperforms all baseline approaches on user-feature preference inference.

The main shortcomings of the baselines are: (i) Each per-user model for NB, LR, and RF leverages information specific to that user only—i.e., her movie ratings, and the corresponding movie features—, and thus suffers from the sparsity problem; (ii) MF falls in the collaborative space, but does not leverage information about the movie features, and operates on a latent feature space; content-based MF also partially relies on latent movie features, which again prohibits its use in movie design. The heterogeneous entity recommendation approach infers the movie preferences, but not the feature preferences, and, thus, cannot be used for movie design either. In contrast, our proposed method, PNP, solves the MD problem by exploiting the latent similarities between users (collaborative), movies, and features, and operates explicitly on the movie features, thus directly enabling us to use its output to design movies.

Table 4: PNP outperforms all the baselines. We report the prediction accuracy (avg AUC and its std in parentheses). α, β, γ are the path combination parameters, and δ is the rating-reweighing parameter.

Method	Avg AUC over 5 folds
Naive Bayes	0.5340 (0.0002)
Logistic Regression	0.6621 (0.1369)
Random Forests	0.6418 (0.1390)
Matrix Factorization	0.6396 (0.0825)
Content-based Matrix Factorization	0.7043 (0.1420)
Heterogeneous Entity Recommendation	0.5998 (0.0112)
PNP	
$(\alpha, \beta, \gamma, \delta) = (0.7, 0.1, 0.2, 1)$	0.9146 (0.0043)
$(\alpha, \beta, \gamma, \delta) = (0.5, 0.2, 0.3, 1)$	0.9143 (0.0043)
$(\alpha, \beta, \gamma, \delta) = (0.3, 0.2, 0.5, 1)$	0.9132 (0.0042)
$(\alpha, \beta, \gamma, \delta) = (0.5, 0.2, 0.3, 0.5)$	**0.9171** (0.0019)
$(\alpha, \beta, \gamma, \delta) = (0.5, 0.2, 0.3, 1.5)$	0.9062 (0.0018)

4.1.2 Q2: Robustness of PNP. **Robustness to the reweighing schema.** To evaluate our reweighing schema in Eq. 1, we learn a PNP classifier per user for different values of the parameter δ. In Figure 5, we give the average AUC (5-fold cross validation) of PNP for using the centered ratings without exponentiation (simple reweighing), and our proposed approach with δ ranging from 0 to 3, and for combination parameters $\alpha = 0.5$, $\beta = 0.2$ and $\gamma = 0.3$. The performance is stable for $\delta = 0 - 1.5$, which corresponds to moderate scaling of the user ratings.

Observation 3. PNP is robust to moderate rescaling of the user-adapted movie ratings.

Figure 5: PNP is robust to moderate rescaling of the centered ratings (x-axis: exponent of proposed reweighing schema in Eq. (1); y-axis: avg AUC over 5-fold cross validation).

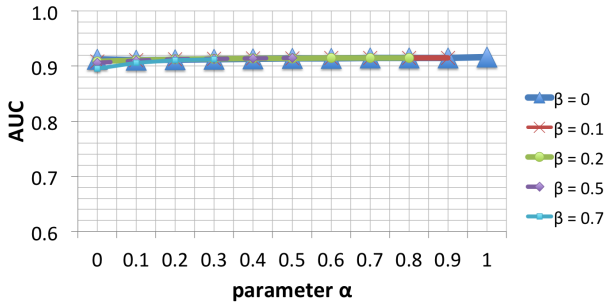


Figure 6: PNP is robust to the combination parameters of the predefined paths (x-axis: varying parameter α; y-axis: avg AUC; different lines for different values of β).

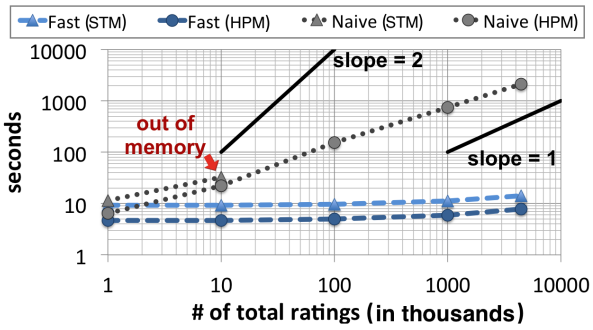


Figure 7: PNP is scalable. The fast approach is faster and needs less space than the naïve one.

Robustness to the combination parameters. As described in Eq. (3), the proposed method, PNP, has three parameters which define the involvement of each predefined path. Next, we explore how these parameters affect the user preference inference. As before, we learn a PNP classifier per user for different values of $\{\alpha, \beta, \gamma\}$, and compute the average AUC and its standard deviation (y-axis) by performing 5-fold cross validation (Figure 6). In this experiment we set $\delta = 1$. We vary the values of α and γ from 0 to 1 with step 0.1 (β is uniquely determined since $\alpha + \beta + \gamma = 1$).

Observation 4. PNP is quite robust to the 'combination' parameters (α, β, γ) of the predefined walks.

The lowest accuracy is obtained when the direct user preferences (2-step path) are ignored, and the highest accuracy when the user-based 4-step path has small participation, likely because this walk 'blurs' the individual preferences by relying on 'similar' users. In practice, these parameters can be set by performing cross validation on the data at hand.

4.1.3 Q3: Scalability. We evaluate the scalability of our method with respect to the number of ratings, which corresponds to the number of non-zeros in the ratings matrix $\mathbf{R}$. For our experiment, we vary the number of ratings from 1000 to 4 435 359, which is the total number of ratings in our dataset. For each number of ratings, we generate five matrices with randomly chosen ratings from the original matrix, we run PNP and report the average and standard deviation of the runtime. For comparison, we ran both the naïve and fast approaches (we give their theoretical complexities in Section 2.3). Figure 7 shows the runtime of the two methods on:

(a) a standard machine (STM): AMD Opteron Processor 854 @ 2.80GHz, 2 cores, 32GB RAM;
(b) a high-performance machine (HPM): AMD Opteron Processor 6282 SE @ 2.60GHz, 16 cores, 264GB RAM.

We note that the naïve approach, which needs to compute a dense $u \times f$ matrix $\mathbf{W}$ in Step 1, runs out of memory for more than 10 000 ratings on the standard machine, while our fast approach is highly scalable and has comparable runtime on both machines. The speedup of the fast approach is due to avoiding the computation of matrix $\mathbf{W}$ (which has over 912 *million* entries for 28 482 users and 32 029 features) and only performing sparse matrix-vector multiplications, based on the observation that only *aggregate* preferences from the target users $\mathcal{U}'$ are needed for the movie design problem under the uniform threshold model (Section 2.3).

Observation 5. The fast approach (sparse matrix-vector multiplications) is up to 100× faster than the naïve approach, and requires less memory.

4.2 Analysis of Step 2: Designing the Movie

4.2.1 Q4: Quantitative Evaluation. Evaluating the second subproblem of MovieDesign is a very challenging task, since there is no ground truth available, and actionable evaluation (i.e., producing a movie based on our method's output) is not feasible. Since no other work in the literature has addressed this problem, we introduce two intuitive baselines (which can be automated, and, thus, do not involve an expert producing a movie):

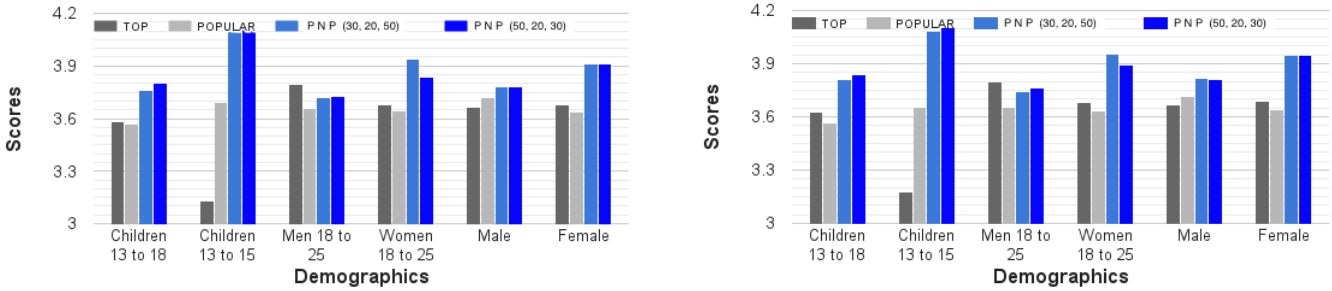
• **Baseline 2.1:** Popular. Using the same capacity constraints as in the optimization problem in Eq. (5), we design a movie by choosing the most popular features for the target audience $\mathcal{U}'$. We assume that the ratings for features are inherited from the movies to which they belong. The $1 \times f$ popularity vector is given by: $\mathbf{p} = \mathbf{v} \cdot \mathbf{F}$, where $\mathbf{v}$ is the vector of total ratings per movie (with elements $v_j = \sum_{i=1}^{u'} \mathbb{1}_{r_{ij}>0}$, where $\mathbb{1}$ is an indicator function), and $\mathbf{F}$ is the binary movie-feature membership matrix.

• **Baseline 2.2:** Top. Applying the same capacity constraints as in Eq. (5), we design a movie by selecting the most highly rated features for the target users $\mathcal{U}'$. We assume that the features inherit their movies' ratings, so the vector of feature ratings can be computed as: $\mathbf{t} = \bar{\mathbf{r}}_\mathbf{j} \cdot \widetilde{\mathbf{F}}$, where $\bar{\mathbf{r}}_\mathbf{j}$ is the vector with the average movie ratings over all users in $\mathcal{U}'$, and $\widetilde{\mathbf{F}}$ is the column-normalized movie-feature membership matrix.

The problem formulation that we introduced in Section 2.2 can handle both capacity and budget constraints during the feature selection for a new movie. Due to the lack of reliable resources that provide the real budgets per actor, director, etc., for the purpose of our experiments, we use only capacity constraints for our method and the two above-mentioned baselines.

Despite the challenges of evaluating this task, we introduce two quantitative measures that leverage the ratings of existing movies (i.e., the available information) to evaluate the new movie designs:

• **kNN**: Given a movie j^*, we find the set $\mathcal{N}$ of its k nearest neighbors (via cosine similarity) and infer its score as $\hat{r}_{j^*} = \frac{1}{|\mathcal{N}|} \sum_{j \in \mathcal{N}} \bar{r}_j$,



(a) **kNN scores per target audience.**

(b) **w-kNN scores per target audience.**

Figure 8: PNP outperforms the TOP and POPULAR movie designs.

where $\bar{\mathbf{r}}$ is the vector with the average movie ratings over all users in $\mathcal{U}'$. We set $k = 20$.

• **w-kNN**: This is the weighted version of kNN, where the movie scores are weighted by their cosine similarity to the given movie j^*, i.e., $\hat{r}_{j^*} = \alpha \sum_{j \in \mathcal{N}} sim(j^*, j) \cdot \bar{r}_j$, where $\alpha = 1/\sum_{j \in \mathcal{N}} sim(j^*, j)$ is a normalization factor.

Results. We applied all methods to the movie dataset (Section 3) in order to design movies (with constraints, e.g. six actors, two directors, two genres) for different target audiences: (i) all women; (ii) all men; (iii) 13- to 15- year-old children; (iv) 13- to 18- year-old children; (v) men 18-25 years old; (vi) women 18-25 years old.

Observation 6. As shown in Figures 8(a) and (b), PNP outperforms the competing baselines for Popular and Top designs and obtains higher kNN and w-kNN scores.

4.2.2 Anecdotes. To highlight how the movie design changes with the age of the target audience, we present the movies designed for men across different age groups. For teenage boys (13-18years), the movie designed has the following composition. Top genres are Martial Arts and Action; top actors chosen are Will Smith (according to IMDB, best known for the Men In Black series) and Adam Sandler (known for Grown Ups); directors are Clint Eastwood (known for Million Dollar Baby) and Spike Lee (known for Malcolm X); producers are Mark Burg (known for Two and a Half Men) and Jack Giarraputo (known for 50 First Dates); studios are Revolution and Warner Brothers. Interestingly, the top chosen genre for a movie for 19-25 year-old men is Romance, with Ned Bellamy (best known for Being John Malkovich) and Alan Rickman (known for the Harry Potter series) as the leading actors, and Ron Howard as the director (known for Cinderella Man). As another example, the movie designed for men of all ages has Martial Arts (the genre for movies like Rush Hour 2) as top-genre, Gary Oldman (known for the Batman series) and Matt Damon (Bourne series) as lead actors, and Steven Spielberg as lead director.

Our model leverages the distinct patterns in the movie interests of different demographics to design movies that specifically cater to those audiences. That said, movie making is a highly creative process and many elements need to come together. Taking such a data-driven approach to designing movies is a step towards the world of targeted and personalized movies.

5 RELATED WORK

We review three main research areas that are related to our work:

Recommendation systems Our work is related to recommendation methods, and specifically group recommendation, which seeks to recommend *existing* items that are likely to match the tastes of multiple users at the same time [15, 30]. The area of recommendation systems is very active with numerous algorithms that leverage the user preferences and similarities [7, 14, 18], or item similarities [24, 32] in order to provide personalized recommendations to the users. The user-based CF approaches suffer from the sparsity problem, especially for new users (cold-start problem), which leads to very low prediction quality and scalability. To overcome these problems, [25] leverages the social interactions and connections between users, [39] proposes a unified approach that combines CF with friendships and memberships, and other methods, known as content-based methods that use side information, such as demographics, and item features (e.g. stylistic visual attributes) [11, 13, 27, 31]. A popular, scalable and accurate approach is matrix factorization [17], which relies on latent factors for both users and items in order to make recommendations. Privacy is always a major concern in online systems, which has also led to privacy-preserving recommendation systems [6, 26, 29]. Despite the similarities with group recommendation, our objective is to design a *new* movie (or, generally, a new product), so that the expected number of endorsers in the intended audience is maximized.

Heterogeneous graphs Heterogeneous networks have become very popular in the recent years, and efforts have focused on adapting and extending approaches intended for homogeneous graphs, such as similarity search [19, 34] and random walk with restarts [4, 28, 41]. Sun et al. [34] introduced the idea of meta-paths for similarity search. In our work, we use meta-paths or predefined paths to perform 'random' walks, with the ultimate goal of inferring user-feature preferences for the MD problem. The heterogeneous entity recommendation approach [38] addresses the problem of top-k entity recommendation by performing matrix factorization on user-movie preference matrices obtained by employing the definition of meta-path similarity [34] (normalized path counts). The goal of [38] is to rank *existing* items by user interest, assuming that all the items that are rated by a user are of interest to her (although she might have disliked some of the items). This is a weaker problem than the one that our work and typical recommender systems

tackle. Moreover, [38] cannot directly infer the user-feature preferences, making it unsuitable for the MD problem. Unlike these works, we infer the user preferences with respect to item *features* by defining random-walk scores over meta-paths and effectively incorporate likes and dislikes, which leads to high accuracy.

Influence Maximization Influence maximization is a fundamental underlying problem in viral marketing [9], the early adoption of products and the dynamics of adoption [23, 33], targeted advertising, and more. The goal is to identify which users to target in order to maximize the adoption of an existing product or service. One of the most influential papers in the area by Kempe, Tardos, and Kleinberg [16] introduced the independent cascades and linear threshold models for user conversion; more efficient models have been proposed since then [10]. In our work, we use a variation of the linear threshold model.

6 CONCLUSIONS

This paper introduces the MovieDesign problem for specific target audiences by leveraging user-movie preferences and movie content. The MD problem that we proposed is complementary to recommendation systems. In addition to contributing a novel formulation of the MD problem as an optimization problem, we introduced a new random walk-based algorithm, PNP on a Positive and Negative graph, which efficiently handles dislikes in the user preferences. We showed that PNP is superior to baseline methods in terms of movie preference predictions. Finally, we applied our method on large, real-world datasets to generate movies for specific audiences, and introduced ways for the qualitative and quantitative evaluation of the designs. Although we tailored our approach to the design of movies, it can be generalized to other products for which reviews and features can be identified.

Future work includes extending our method with other elements of the creative movie process, notably the plot of the movie, the screenplay, and the soundtrack. There are additional signals that can be incorporated, such as the credits order for the actors in the movie as an indicator of their contribution and weights on a movie's genre, e.g. for a movie that is mostly drama yet has an element of romance, we can weigh those two genres unequally.

7 ACKNOWLEDGEMENTS

The authors thank Ali Radha and William Sullivan for their involvement in the evaluation of the proposed approaches, and Yaohua Shi for his helpful feedback on the paper. The authors also thank the anonymous reviewers for their insightful comments.

This material is based upon work supported by the *University of Michigan* and *Northeastern University*.

REFERENCES

[1] Alexander A. Ageev and Maxim Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *J. Comb. Optim.*, 8(3):307–328, 2004.
[2] Aris Anagnostopoulos, Luca Becchetti, Carlos Castillo, Aristides Gionis, and Stefano Leonardi. Online Team Formation in Social Networks. In *WWW*, 2012.
[3] Marc Andreessen. Part 4: The only thing that matters, 2007.
[4] Andrey Balmin, Vagelis Hristidis, and Yannis Papakonstantinou. Objectrank: Authority-based keyword search in databases. In *VLDB*, 2004.
[5] James Bennett and Stan Lanning. The Netflix prize. In *KDD Cup*, 2007.
[6] Smriti Bhagat, Udi Weinsberg, Stratis Ioannidis, and Nina Taft. Recommending with an Agenda: Active Learning of Private Attributes Using Matrix Factorization. In *RecSys*, 2014.
[7] John S. Breese, David Heckerman, and Carl Myers Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *UAI*, 1998.
[8] David Carr. Giving viewers what they want. *New York Times*, 2013.
[9] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD*, 2010.
[10] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *KDD*, 2009.
[11] Yashar Deldjoo, Mehdi Elahi, Paolo Cremonesi, Franca Garzotto, Pietro Piazzolla, and Massimo Quadrana. Content-based video recommendation system based on stylistic visual features. *J Data Semant*, 2016.
[12] Stephan Dernbach, Nina Taft, Jim Kurose, Udi Weinsberg, Christophe Diot, and Azin Ashkan. Cache Content-Selection Policies for Streaming Video Services. In *INFOCOM*, 2016.
[13] Quanquan Gu, Jie Zhou, and Chris Ding. Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs. In *SDM*, 2010.
[14] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative Filtering for Implicit Feedback Datasets. In *ICDM*, 2008.
[15] Anthony Jameson and Barry Smyth. In *The Adaptive Web*, chapter Recommendation to Groups. Springer-Verlag, 2007.
[16] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.
[17] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.
[18] Danai Koutra, Tai-You Ke, U Kang, Duen Horng Chau, Hsing-Kuo Kenneth Pao, and Christos Faloutsos. Unifying Guilt-by-Association Approaches: Theorems and Fast Algorithms. In *ECML PKDD*, 2011.
[19] Danai Koutra, Neil Shah, Joshua T. Vogelstein, Brian Gallagher, and Christos Faloutsos. Deltacon: Principled massive-graph similarity function with attribution. *ACM TKDD*, 10(3):28:1–28:43, 2016.
[20] Theodoros Lappas, Kun Liu, and Evimaria Terzi. Finding a Team of Experts in Social Networks. In *KDD*, 2009.
[21] Jessica Leber. "House of Cards" and our future of algorithmic programming. *MIT Technology Review*, 2013.
[22] Liangyue Li, Hanghang Tong, Nan Cao, Kate Ehrlich, Yu-Ru Lin, and Norbou Buchler. Replacing the Irreplaceable: Fast Algorithms for Team Member Recommendation. In *WWW*, 2015.
[23] Yibin Lin, Agha Ali Raza, Jay Yoon Lee, Danai Koutra, Roni Rosenfeld, and Christos Faloutsos. Influence Propagation: Patterns, Model and a Case Study. In *PAKDD*, 2014.
[24] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Comput.*, 7(1):76–80, 2003.
[25] Hao Ma, Haixuan Yang, Michael R. Lyu, and Irwin King. SoRec: Social Recommendation Using Probabilistic Matrix Factorization. In *CIKM*, 2008.
[26] Frank McSherry and Ilya Mironov. Differentially Private Recommender Systems: Building Privacy into the Netflix Prize Contenders. In *KDD*, 2009.
[27] Prem Melville, Raymod J. Mooney, and Ramadass Nagarajan. Content-boosted Collaborative Filtering for Improved Recommendations. In *AAAI*, 2002.
[28] Zaiqing Nie, Yuanzhi Zhang, Ji-Rong Wen, and Wei-Ying Ma. Object-level ranking: bringing order to web objects. In *WWW*, 2005.
[29] Valeria Nikolaenko, Stratis Ioannidis, Udi Weinsberg, Marc Joye, Nina Taft, and Dan Boneh. Privacy-preserving Matrix Factorization. In *CCS*, 2013.
[30] Mark O'Connor, Dan Cosley, Joseph A. Konstan, and John Riedl. Polylens: A recommender system for groups of users. ECSCW, 2001.
[31] Royi Ronen, Noam Koenigstein, Elad Ziklik, and Nir Nice. Selecting content-based features for collaborative filtering recommenders. In *RecSys*, 2013.
[32] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based Collaborative Filtering Recommendation Algorithms. In *WWW*, 2001.
[33] Yaron Singer. How to win friends and influence people, truthfully: Influence maximization mechanisms for social networks. In *WSDM*, 2012.
[34] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB*, 2011.
[35] Technicolor. MGO. http://www.mgo.com/, 2014.
[36] Madeleine Udell and Stephen Boyd. Maximizing a sum of sigmoids, 2013.
[37] Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.
[38] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. Personalized entity recommendation: A heterogeneous information network approach. In *WSDM*, 2014.
[39] Quan Yuan, Li Chen, and Shiwan Zhao. Factorization vs. Regularization: Fusing Heterogeneous Social Relationships in Top-n Recommendation. In *RecSys*, 2011.
[40] Reza Zafarani and Huan Liu. Social computing data repository at ASU, 2009.
[41] Jing Zhang, Jie Tang, Bangyong Liang, Zi Yang, Sijie Wang, Jingjing Zuo, and Juanzi Li. Recommendation over a heterogeneous social network. In *WAIM*, 2008.