# HyPER: A Flexible and Extensible Probabilistic Framework for Hybrid Recommender Systems

Pigi Kouki
UC Santa Cruz
pkouki@soe.ucsc.edu

Shobeir Fakhraei
University of Maryland
shobeir@cs.umd.edu

James Foulds
UC Santa Cruz
jfoulds@ucsc.edu

Magdalini Eirinaki
San Jose State University
magdalini.eirinaki@sjsu.edu

Lise Getoor
UC Santa Cruz
getoor@soe.ucsc.edu

## ABSTRACT

As the amount of recorded digital information increases, there is a growing need for flexible recommender systems which can incorporate richly structured data sources to improve recommendations. In this paper, we show how a recently introduced statistical relational learning framework can be used to develop a generic and extensible hybrid recommender system. Our hybrid approach, HyPER (HYbrid Probabilistic Extensible Recommender), incorporates and reasons over a wide range of information sources. Such sources include multiple user-user and item-item similarity measures, content, and social information. HyPER automatically learns to balance these different information signals when making predictions. We build our system using a powerful and intuitive probabilistic programming language called probabilistic soft logic [1], which enables efficient and accurate prediction by formulating our custom recommender systems with a scalable class of graphical models known as hinge-loss Markov random fields. We experimentally evaluate our approach on two popular recommendation datasets, showing that HyPER can effectively combine multiple information types for improved performance, and can significantly outperform existing state-of-the-art approaches.

## Keywords

Hybrid recommender systems, graphical models, probabilistic programming, probabilistic soft logic.

## 1. INTRODUCTION

Recent work on hybrid recommender systems has shown that recommendation accuracy can be improved by combining multiple data modalities and modeling techniques within a single model [2, 3, 4, 5, 6]. Existing hybrid recommender systems are typically designed for a specific problem domain, such as movie recommendations, and are limited in their ability to generalize to other settings or make use of any further information. As our daily lives become increasingly digitally connected, the list of data sources available for recommendations continues to grow. There is a need for general-purpose, extensible frameworks that can make use of arbitrary data modalities to improve recommendation.

The challenge of custom model-building has been extensively studied in the fields of probabilistic programming [7] and statistical relational learning (SRL) [8], which provide programming language interfaces for encoding knowledge and specifying models. Probabilistic programs can be used to encode graphical models for reasoning with graph-structured probabilistic dependencies. Graphical models are a natural approach to recommendations given that the user-item rating matrix can be interpreted as a graph, with weighted edges between users and items corresponding to the respective ratings [4].

In modern recommendation contexts, a bipartite user-item graph is insufficient to represent all available information, such as user-user and item-item similarity, content, social information, and metadata. For example, neighborhood-based collaborative filtering techniques can be interpreted as predicting ratings based on an extension of the user-item graph with additional edges between pairs of similar users or similar items (Figure 1). We need a more general representation to reason over this richly structured information.

In this paper, we propose a general hybrid recommender framework, called HyPER (HYbrid Probabilistic Extensible Recommender), which leverages the flexibility of probabilistic programming in order to build adaptable and extensible hybrid recommender systems which reason over complex data. In particular, we use a modeling language called *probabilistic soft logic* (PSL) [9]. PSL is especially well-suited to collaborative-filtering based recommendation graphs as it is able to fuse information from multiple sources and it was originally designed as a flexible framework for reasoning and combining similarities [10]. It provides a general declarative framework for combining entity similarities, attribute similarities, and information from additional sources including the predictions of other algorithms. The models defined by PSL programs, called *hinge-loss Markov random fields* (HL-MRFs), are amenable to efficient and scalable inference, which is crucial in a recommendation context.

Our contributions include (1) a general and extensible hybrid recommender system with a probabilistic programming interface, (2) a method for learning how to balance the different input signals in the hybrid system, and (3) extensive experimental studies using several information sources which validate the performance of the proposed framework and highlight contribution of each source to the final pre-



*RecSys'15 September 16–20, 2015, Vienna, Austria*



ISBN 978-1-4503-3692-5/15/09…$15.00

DOI: http://dx.doi.org/10.1145/2792838.2800175

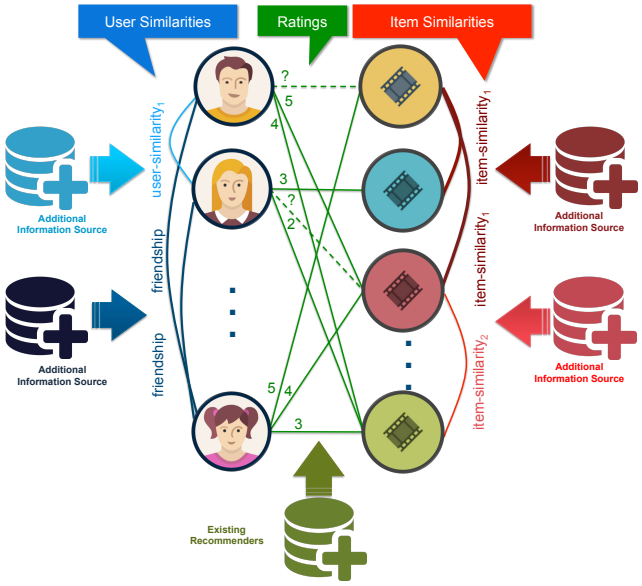


**Figure 1: Example recommendation graph.**

diction. To the best of our knowledge, our proposed HyPER framework is the first which provides a mechanism to extend the system by incorporating and reasoning over currently unspecified additional information types and similarity measures. We evaluate our system on two rich datasets from the local business and music recommendation domains (Yelp and Last.fm) comparing our model to state-of-the-art recommendation approaches. Our results show that HyPER is able to effectively combine multiple information sources to improve recommendations, resulting in significantly improved performance over the competing methods in both datasets.

## 2. BACKGROUND

Recommender systems play a significant role in many everyday decision-making processes which affect the quality of our lives, from the restaurant we have lunch at, to the hotel for our vacation, to the music we listen to. Traditional recommender systems primarily leverage underlying similarities between users and items in order to make predictions based on observed ratings. Content-based filtering (CB) approaches compute these similarities by using features extracted from content to build user profiles, which are compared with content features of items. While content-based approaches can recommend newly added items, they are limited by a lack of serendipity. The recommendations are limited to the user's known likes and do not generally include items out of the user's (recorded) comfort zone [11].

Collaborative filtering (CF) techniques address this by identifying similar users or items based on their rating patterns instead of content, using methods such as neighborhood-based approaches and matrix factorization models. However, collaborative filtering methods typically do not perform well in "cold-start" settings, where there are few ratings for a user or an item [12]. Moreover, pure rating-based collaborative filtering approaches cannot take advantage of data which may be available in addition to ratings.

To address these shortcomings, hybrid recommender systems (HRSs) were introduced, combining content-based and collaborative-filtering techniques (e.g. [2, 3, 4]). HRS techniques can improve performance over content-based and collaborative filtering methods alone, especially in the case where the ratings matrix is sparse [2]. However, existing HRSs have their own limitations. First, they are problem- and data-specific. Each HRS is typically motivated by a specific problem domain (e.g. movie recommendations) and the solution is fine-tuned to solve a specific problem with datasets of specific characteristics. Hence, HRSs typically cannot be generalized to different problem domains or input data, or be easily expanded to incorporate knowledge from richer datasets.

As the web has evolved into a participatory, user-driven platform, additional information is increasingly becoming available. Users form social networks, give verbal feedback on items via reviews, endorse or down-vote items or other users, form trust relationships, "check-in" at venues, and perform many other social actions that may potentially be leveraged to better understand users in order to improve recommendations. A flexible and extensible hybrid recommender system which can make use of this wealth of information is increasingly important.

The remainder of the paper is structured as follows. In Section 3 we introduce HyPER, a general hybrid recommendation framework which is extensible and customizable using a probabilistic programming interface. We systematically evaluate our framework in Section 4, and place our system in the context of related work in Section 5. Finally, we conclude with a discussion in Section 6.

## 3. PROPOSED APPROACH

We propose HyPER, a general hybrid framework that combines multiple different sources of information and modeling techniques into a single unified model. HyPER offers the capability to extend the model by incorporating additional sources of information as they become available. Our approach begins by viewing the recommendation task as a bipartite graph, where users $\mathcal{U}$ and items $\mathcal{I}$ are the vertices, and ratings are edges between users and items [4]. Using PSL [1], a flexible statistical relational learning system with a probabilistic programming interface, this graph is then augmented to construct a probabilistic graphical model with additional edges to encode similarity information, predicted ratings, content and social information, and metadata. We then train the graphical model to learn the relative importance of the different information sources in the hybrid system, and make predictions for target ratings, using graphical model learning and collective inference techniques.

Figure 1 shows an overview of our modeling approach. In the figure, items and users are nodes, and green edges represent the ratings that users gave to items, with edge weights corresponding to the rating values. The goal is to predict the edge weights for unobserved edges, denoted as dashed lines. Neighborhood-based approaches find the $k$ most similar users or similar items, and use their ratings to make these predictions. In our graph-based representation, we interpret these $k$-nearest neighbor relationships as $k$ edges which are added to the graph. In Figure 1, blue edges encode user similarities and red edges correspond to item similarities.

We can further encode additional sources of information and outputs of other recommendation algorithms within this graph-based representation in a similar way, i.e. in the form of additional links or nodes. For instance, latent factor methods identify latent representations which can be used

to augment the graph with weighted edges encoding predictions of user-item ratings based on the latent space. The latent representations can also be used to construct additional user-user and item-item edges by identifying similar users and similar items in the latent space. Content information and metadata, such as demographics and time information, can be incorporated in the graph representation by identifying further similarity links, or by adding nodes with attribute values, and edges to associate these values with users and items. Furthermore, social information from digital social media is inherently relational, and can readily be incorporated into a graph-based representation.

Having encoded all available information in the graph, the next step is to reason over this graph to predict unobserved user-item rating edges. We view the prediction task as inference in a graphical model, the structure of which is defined by our graph representation. As scalability is important for recommendation tasks in practice, we use a hinge-loss Markov random field (HL-MRF) formulation [9]. HL-MRFs are highly scalable as they admit exact inference by way of efficient parallel algorithms. In the next section we briefly review HL-MRFs. We then describe our unified recommender system modeling framework in detail in Section 3.2, and we show how to learn the relative importance of the information sources for our hybrid model in Section 3.3.

## 3.1 Hinge-loss Markov Random Fields

Hinge-loss Markov random fields (HL-MRFs) [9] are a general class of conditional probabilistic models over continuous random variables which admit tractable and efficient inference. The key to the tractability of these models is the use of hinge-loss feature functions. More formally, a hinge-loss Markov random field defines a conditional probability density function over random variables $\mathbf{Y}$ conditioned on $\mathbf{X}$,

$$P(\mathbf{Y}|\mathbf{X}) \propto \exp\Big(-\sum_{j=1}^{m} w_j \phi_j(\mathbf{Y},\mathbf{X})\Big) \ , \tag{1}$$

where $\phi_j$ is a *hinge-loss* potential function, of the form

$$\phi_j(\mathbf{Y},\mathbf{X}) = (max\{\ell_j(\mathbf{X},\mathbf{Y}),0\})^{p_j} \ . \tag{2}$$

Here, $\ell$ is linear function of $\mathbf{X}$ and $\mathbf{Y}$, and $p_j \in \{1,2\}$ optionally squares the potential. The variables in $\mathbf{X}$ and $\mathbf{Y}$ are in the unit interval [0,1]. Each $\phi_j$ is associated with a weight $w_j$ which determines its importance in the model. We learn the weights from the data, as discussed later in Section 3.3. Note that Equation 1 is log-concave in $\mathbf{Y}$, so maximum a posteriori (MAP) inference to find the optimal $\mathbf{Y}$ in HL-MRFs can be solved exactly via convex optimization. We use the alternating direction method of multipliers (ADMM) approach of Bach et al. [9] to perform this optimization efficiently and in parallel.

HL-MRFs can be specified using a probabilistic programming language called *Probabilistic Soft Logic (PSL)* [1], and this is the language we use to specify our unified recommendation framework. PSL is a declarative first-order logical language where logical rules are composed of continuous relaxations of Boolean logical operators. These rules define templates for hinge-loss potential functions, which are instantiated to construct an HL-MRF model. For example, $a \Rightarrow b$ corresponds to the hinge function $\max(a-b,0)$, and $a \wedge b$ corresponds to $\max(a+b-1,0)$. We refer the reader to [1] for a detailed description of PSL operators.

To illustrate PSL in a movie recommendation context, the following rule encodes that users tend to rate movies of their preferred genres highly:

$$LikesGenre(U,G) \wedge IsGenre(M,G) \Rightarrow Rating(U,M) \ ,$$

where $LikesGenre(U,G)$ is a binary observed predicate, $IsGenre(M,G)$ is a continuous observed predicate in the interval $[0,1]$ capturing the affinity of the movie to the genre, and $Rating(U,M)$ is a continuous variable to be inferred, which encodes the star rating as a number between 0 and 1, with higher values corresponding to higher star ratings. For example, we could instantiate $U = Jim$, $G = classics$ and $M = Casablanca$. This instantiation results in a hinge-loss potential function in the HL-MRF,

$$\begin{aligned}\max(&LikesGenre(Jim, classics)\\ &+ IsGenre(Casablanca, classics)\\ &- Rating(Jim, Casablanca) - 1, 0) \ .\end{aligned}$$

## 3.2 Hybrid Framework

The strengths of the HyPER framework include the ability to extensibly incorporate multiple sources of information in a unified hybrid recommendation model, as well as learning how to balance these signals from training data. HyPER models are specified using a collection of PSL rules which encode graph-structured dependency relationships between users, items, ratings, content and social information. Additionally, the model provides the flexibility to incorporate prior predictions, such as mean-centering predictors and the results of other recommendation algorithms. In what follows, we present the rules that define the HL-MRF model for the core of the HyPER framework. We emphasize that while this set of rules covers a breadth of input sources, the model can be readily extended to incorporate other sources of information such as time, implicit feedback, and social interactions, with the introduction of new PSL rules. Moreover, additional similarity measures and recommendation algorithms can straightforwardly be included with analogous rules. HyPER builds upon the ideas of Fakhraei et al. [13], which is an important precursor to this work.

### 3.2.1 User-based Collaborative Filtering

Motivated by the basic principles of the neighborhood-based approach, we can define PSL rules of this form:

$$SimilarUsers_{sim}(u_1,u_2) \wedge Rating(u_1,i) \Rightarrow \ Rating(u_2,i) \ .$$

This rule captures the intuition that similar users give similar ratings to the same items. The predicate $Rating(u,i)$ takes a value in the interval $[0,1]$ and represents the normalized value of the rating that a user $u$ gave to an item $i$, while $SimilarUsers_{sim}(u_1,u_2)$ is binary, with value 1 iff $u_1$ is one of the $k$-nearest neighbors of $u_2$. The similarities can be calculated with any similarity measure $sim$, as we will describe in Section 3.2.3. The above rule represents a template for hinge functions which reduce the probability of predicted ratings as the difference between $Rating(u_2,i)$ and $Rating(u_1,i)$ increases, for users that are neighbors.

### 3.2.2 Item-based Collaborative Filtering

Similarly, we can define PSL rules to capture the intuition of item-based collaborative filtering methods, namely that similar items should have similar ratings from the same users:

$$SimilarItems_{sim}(i_1,i_2) \wedge Rating(u,i_1) \Rightarrow \ Rating(u,i_2) \ .$$

The predicate $SimilarItems_{sim}(i_1, i_2)$ is binary, with value 1 iff $i_1$ is one of the $k$-nearest neighbors of $i_2$ (using similarity measure $sim$), while $Rating(u, i)$ represents the normalized value of the rating of user $u$ to item $i$, as discussed above.

### 3.2.3 Combining Collaborative Filtering Measures

By including both types of rules described in Sections 3.2.1 and 3.2.2 we can define an HL-MRF model that combines user-based and item-based techniques to predict ratings. There exist many measures available to compute similarities between entities for user-based and item-based methods, and these different measures capture different notions of similarity. For instance, in neighborhood-based approaches, vector-based similarity measures are broadly used, whereas in latent factor approaches other similarities, applicable to the low dimensional space, are preferred. While most existing recommender systems are designed to use a single similarity measure, HyPER allows for the simultaneous incorporation of multiple similarity measures, and can automatically adjust the importance of each based on training data.

In this instantiation of our HyPER framework we use the most popular similarity measures in the neighborhood-based recommendations literature [4]. More specifically, we apply Pearson's correlation and cosine similarity measures to calculate similarities between users and items; for the items we additionally apply the adjusted cosine similarity measure. To incorporate matrix-factorization collaborative filtering, and inspired by Hoff et al. [14], we compute similar users and items in the low-dimensional latent space using two popular distance measures in that space, namely, cosine and Euclidean. The user similarities are identified using the following rules:

$$\begin{aligned}
SimilarUsers_{cosine}(u_1, u_2) \wedge Rating(u_1, i) &\Rightarrow \quad Rating(u_2, i) \\
SimilarUsers_{pearson}(u_1, u_2) \wedge Rating(u_1, i) &\Rightarrow \quad Rating(u_2, i) \\
SimilarUsers_{\substack{latent \\ cosine}}(u_1, u_2) \wedge Rating(u_1, i) &\Rightarrow \quad Rating(u_2, i) \\
SimilarUsers_{\substack{latent \\ euclidean}}(u_1, u_2) \wedge Rating(u_1, i) &\Rightarrow \quad Rating(u_2, i) \ .
\end{aligned}$$

Analogous rules are introduced to identify similar items, but are omitted due to space limitations. As noted earlier, this initial set of similarity measures can be readily expanded by adding the corresponding rules, in the same form as above.

### 3.2.4 Mean-Centering Priors

Each individual user considered in a recommender system has her own biases in rating items (e.g. some users tend to be stricter than others). Moreover, each item's rating is influenced by its overall quality and popularity (e.g. a popular blockbuster may get higher ratings on average than a low-budget movie). To address such biases, a recommender system needs to incorporate a normalization mechanism, both per user, and per item. Using mean-centering normalization for neighborhood-based approaches, or including intercept terms in probabilistic latent factor models, addresses this issue and generally improves performance [4]. In our HyPER framework we encode this intuition with rules that encourage the ratings to be close to the average, per-user and per-item:

$$\begin{aligned}
AverageUserRating(u) &\Rightarrow \quad Rating(u, i) \\
\neg AverageUserRating(u) &\Rightarrow \neg Rating(u, i) \\
\\
AverageItemRating(i) &\Rightarrow \quad Rating(u, i) \\
\neg AverageItemRating(i) &\Rightarrow \neg Rating(u, i) \ .
\end{aligned}$$

The predicate $AverageUserRating(u)$ represents the average of the ratings over the set of items that user $u$ provided in the training set. Similarly, $AverageUserRating(i)$ represents the average of the user ratings an item $i$ has received. The pair of PSL rules per-user and per-item corresponds to a "V-shaped" function centered at the average rating, which penalizes the predicted rating for being different in either direction from this average.

In order to capture cases where we have no information about a user or an item, we use a general prior rating centered at the average value of all of the ratings in the system (i.e. the average over all items rated by all users). We encode this prior with the following rules:

$$\begin{aligned}
PriorRating &\Rightarrow \quad Rating(u, i) \\
\neg PriorRating &\Rightarrow \neg Rating(u, i) \ .
\end{aligned}$$

The real-valued predicate $PriorRating$ represents the average of all of the ratings.

### 3.2.5 Using Additional Sources of Information

Incorporating other sources of information pertaining to the items, the users, and the respective ratings to our framework is straightforward. In the present instantiation of our framework, we use the content of the items to find similar items:

$$SimilarItems_{Content}(i_1, i_2) \wedge Rating(u, i_1) \Rightarrow \quad Rating(u, i_2) \ .$$

In this rule, the predicate $SimilarItems_{Content}(i_1, i_2)$ represents items that have similar content-based features (e.g. in the movie recommendation domain such features are the genre, actor, director, etc.), instead of similar ratings.

The HyPER framework can also incorporate social information, when this is available. For instance, in the present instantiation of the system, we leverage social network friendship links as follows:

$$Friends(u_1, u_2) \wedge Rating(u_1, i) \Rightarrow \quad Rating(u_2, i) \ .$$

Note that our framework is flexible and can incorporate many other sources of information that are available. For instance, we can leverage demographic information by computing similarity neighborhood relationships in demographic feature space and employing the rule:

$$SimilarUsers_{Demo}(u_1, u_2) \wedge Rating(u_1, i) \Rightarrow \quad Rating(u_2, i) \ .$$

### 3.2.6 Leveraging Existing Recommendation Algorithms

Every recommendation algorithm has strengths and weaknesses which may depend on data-specific factors such as the degree of sparsity or the shape of the data matrix. This imposes a big limitation in the recommendation process, as choosing one algorithm as the core of a recommender system limits its strength to a particular set of domains. In this work, our motivation is to provide a flexible framework that can be used as-is to generate accurate recommendations for any domain and data regime. Therefore, instead of selecting a single recommendation algorithm, we propose to incorporate the predictions from different methods into our unified model. These predictions are further augmented with any other available information, using the rules discussed above. For example, the predictions from matrix factorization (optimizing regularized squared error via stochastic gradient descent) (MF), Bayesian probabilistic matrix factorization

(BPMF) [15], and item-based collaborative filtering can be incorporated in the model via the following rules:

$$\begin{aligned} Rating_{MF}(u,i) &\Rightarrow \ Rating(u,i) \\ \neg Rating_{MF}(u,i) &\Rightarrow \neg Rating(u,i) \end{aligned}$$

$$\begin{aligned} Rating_{BPMF}(u,i) &\Rightarrow \ Rating(u,i) \\ \neg Rating_{BPMF}(u,i) &\Rightarrow \neg Rating(u,i) \end{aligned}$$

$$\begin{aligned} Rating_{\substack{item \\ based}}(u,i) &\Rightarrow \ Rating(u,i) \\ \neg Rating_{\substack{item \\ based}}(u,i) &\Rightarrow \neg Rating(u,i) \ . \end{aligned}$$

Additional algorithms can be easily incorporated in a similar manner.

## 3.3 Balancing the Information Sources

An important task of any hybrid recommender system is to trade off and balance the different information sources according to their informativeness for predicting ratings. Each of the first-order rules introduced above corresponds to a different information source in our hybrid model, and is associated with a non-negative weight $w_j$ in Equation 1. These weights determine the relative importance of the information sources, corresponding to the extent to which the corresponding hinge function $\phi_j$ alters the probability of the data under Equation 1, with higher weight $w_j$ corresponding to a greater importance of information source $j$. For each rule we learn a weight using Bach et al. [9]'s approximate maximum likelihood weight learning algorithm for templated HL-MRFs. The algorithm approximates a gradient step in the conditional likelihood,

$$\frac{\partial log P(\mathbf{Y}|\mathbf{X})}{\partial w_j} = \mathbb{E}_w[\phi_j(\mathbf{Y},\mathbf{X})] - \phi_j(\mathbf{Y},\mathbf{X}) \ , \qquad (3)$$

by replacing the intractable expectation with the MAP solution based on $w$, which can be rapidly solved using ADMM.

## 3.4 Scaling to Large Datasets

Due to the hinge-loss formulation, inference and learning are relatively scalable via ADMM, which can be performed in parallel. The UMD/UCSC implementation of PSL uses a single-machine multi-threaded ADMM algorithm, and we use this implementation for our experiments on datasets with around 100,000 ratings. For deployment in industrial applications with millions of users, items, and ratings, the main bottleneck for scalability is memory to store the ground model. This can be addressed simply in the current implementation of PSL by dividing the graph into densely connected subgraphs, e.g. a subgraph per city, that are inferred independently and in parallel on different machines. Alternatively, a fully distributed implementation of ADMM would straightforwardly facilitate the scaling of HyPER to the big data setting via model parallelism.

# 4. EXPERIMENTAL VALIDATION

In this section we evaluate our HyPER framework with comparison to state-of-the-art recommender algorithms. We report experimental results on two popular datasets for both the complete hybrid model and for each individual component of our hybrid models.[1]

[1]Code is available at https://github.com/pkouki/recsys2015

**Table 1: Dataset Description**

| Dataset | Yelp | Last.fm |
|---|---|---|
| **No. of users** | 34,454 | 1,892 |
| **No. of items** | 3,605 | 17,632 |
| **No. of ratings** | 99,049 | 92,834 |
| **Content** | 514 business categories | 9,719 artist tags |
| **Social** | 81,512 friendships | 12,717 friendships |
| **Sparsity** | 99.92% | 99.72% |

## 4.1 Datasets and Evaluation Metrics

For our experimental evaluation we used the Yelp academic dataset and the Last.fm dataset.[2][3] Our goal with Yelp is to recommend local businesses to users by predicting the missing ratings of businesses based on previous ratings. For our experiments, we used all businesses, users, and ratings from Scottsdale, Arizona, one of the largest cities in the dataset. Since we employ user and item similarities as well as social information, it makes sense to focus those relationships within the subgroup of the businesses of one physical location. Additionally, we used the categories of each business as content information and the explicit user friendships provided as social information. Yelp users give ratings from 1 to 5 stars, which we linearly scaled into the [0,1] range that PSL operates over for the purposes of our model.

For the Last.fm dataset our goal is to recommend artists to users. As Last.fm does not provide explicit user-artist ratings we leverage the number of times a user has listened to an artist to construct implicit ratings. We use a simple model-based approach, where the repeated-listen counts for each user across artists are modeled with a negative-binomial distribution. We used this distribution as it is appropriate for count data where the sample variance is greater than the sample mean, which is typically the case for Last.fm. For each user, we fit a negative binomial to their counts via maximum likelihood estimation, and we calculate the user's implicit rating for an artist as the cumulative distribution function (CDF) of the distribution, evaluated at the artist's count. This corresponds to the proportion of hypothetical artists that a user would listen to less than the given artist, under the model. The Last.fm dataset also includes tags on artists that we use for content-based information, as well as user friendship data that we use for social recommendation.

We deliberately selected two datasets with a similar total number of ratings but a different ratio of users to items. Different recommendation methods may perform better with more users than items or vice versa, and hybrid systems must account for this. We provide the summary statistics of the two datasets in Table 1.

To learn the appropriate balance between information sources for HyPER, i.e. to learn the weights of each rule in the model, we train using the approximate maximum likelihood method described in Section 3.3, with 20% of the training folds treated as the prediction target variables $\mathbf{Y}$. During testing, we performed MAP inference to make predictions using ADMM. We report the root mean squared error (RMSE) and the mean absolute error (MAE). We compute these metrics by performing 5-fold cross-validation and reporting the average cross-validated error.

[2]https://www.yelp.com/academic_dataset
[3]http://grouplens.org/datasets/hetrec-2011/

Table 2: Overall Performance of Different Recommender Systems on Yelp and Last.fm.

| | | Yelp | | Last.fm | |
|---|---|---|---|---|---|
| | Model | RMSE (SD) | MAE (SD) | RMSE (SD) | MAE (SD) |
| Base models | Item-based | 1.216 (0.004) | 0.932 (0.001) | 1.408 (0.010) | 1.096 (0.008) |
| | MF | 1.251 (0.006) | 0.944 (0.005) | 1.178 (0.003) | 0.939 (0.003) |
| | BPMF | 1.191 (0.003) | 0.954 (0.003) | 1.008 (0.005) | 0.839 (0.004) |
| Hybrid models | Naive hybrid (averaged predictions) | 1.179 (0.003) | 0.925 (0.002) | 1.067 (0.004) | 0.857 (0.004) |
| | BPMF-SRIC | 1.191 (0.004) | 0.957 (0.004) | 1.015 (0.004) | 0.842 (0.004) |
| | HyPER | **1.173** (0.003) | **0.917** (0.002) | **1.001** (0.004) | **0.833** (0.004) |

## 4.2 Experimental Results

We report overall results with comparison to a selection of competing algorithms in Table 2, and show more detailed results for the individual components of our hybrid models in Table 3. The following sections discuss these results.

### 4.2.1 *Overall Performance Comparison*

We study the performance of HyPER in comparison to several state-of-the-art models. We considered the following baselines:

- **Item-based:** The method in Equation 4.16 from [4], using Pearson's correlation with a mean-centering correction, as implemented in Graphlab.[4]
- **Matrix factorization (MF):** MF optimizing for regularized squared error using stochastic gradient descent [12], as implemented in Graphlab.
- **Bayesian probabilistic matrix factorization (BPMF):** The Bayesian variant of probabilistic matrix factorization, trained using Gibbs sampling [15].
- **Naive hybrid:** A simple hybrid approach where the predictions of the above models are averaged.
- **BPMF with social relations and items' content (BPMF-SRIC):** A hybrid model that extends BPMF with social and content information [5].

The performance of our model is statistically significantly better than the baselines at $\alpha = 0.05$ for both datasets and evaluation metrics when using paired t-test. We denote with bold the numbers that are statistically significantly better. These results confirm our initial intuition that by incorporating a wide variety of information sources and balancing them appropriately, the HyPER framework manages to perform very well with rich and diverse datasets. We explore HyPER components in more detail in the following section.

### 4.2.2 *Performance per Information Type*

For each type of information, we further evaluated our approach by building simple HyPER models with each rule individually, and comparing these to combined hybrid sub-models comprising all of the corresponding rules of that type. Each sub-model also included the corresponding mean-centering rules (e.g. the user-average rating rule for the user-based models). To balance the effect of each rule, we performed weight learning within each training fold to learn rule weights. We report the results in Table 3. The results show that for each information type, the HyPER model which combines all of the corresponding components performs significantly better than each component, considered in isolation. We denote with bold the cases where the performance of each HyPER model is statistically significantly better than all the individual models in the same category at $\alpha = 0.05$ using paired t-test. The final HyPER model shown in the last line, which combines all of the available information into a single hybrid model, also performs statistically significantly better than all sub-models and baselines.

**Mean-Centering Priors:** We created simple HyPER models using only the average rating of each user, or the average rating of each item, or the average overall rating, as well as a combined model. In the case of Yelp, the item-average model had a lower error compared to the user average rule, while the opposite was true for Last.fm (Table 3(a)). This may be because the ratio of users to items is different in the two datasets. The combined model performed better than the individual models in both datasets.

**Neighborhood-Based Collaborative Filtering:** We constructed individual models based on the similarities described in Section 3.2.1. The number of neighbors is typically set to between 20 and 50 in the literature [4], and so we used 50 neighbors for users/item in all experiments. We also employed a mean-centered approach by providing each of these models with the corresponding average-rating mean-centering rules (e.g. the average user-rating rule for user-based collaborative filtering). As in the previous experiment, user-based techniques perform poorly on Yelp, but have better performance on Last.fm (Tables 3(b) and 3(c)). The opposite is true for the item-based techniques, which perform poorly on Last.fm, but better on Yelp. The performance varied between the different similarity measures, with distances computed in the latent space usually performing the best individually. Again, the HyPER combination of all similarity measures improves performance.

**Additional Sources of Information:** We constructed individual and hybrid models using friendship information, as well as content similarity between items based on the business category and the tags of artists for Yelp and Last.fm respectively. We used Jaccard similarity for both datasets. In each sub-model we also provided additional rules for mean centering using both average user and item ratings. Content and friendship information help performance in both datasets, and the model that combines both content and social information matched and often improved on the best individual models' performance (Table 3(d)).

**Leveraging Existing Algorithms:** As discussed in section 3.2.6, our framework is able to combine predictions from a number of different models. In Table 3(e) we show the performance of three baseline recommenders, and in the fourth line we present the results of a HyPER ensemble which com-

[4] http://www.dato.com

Table 3: Performance of HyPER sub-models on Yelp and Last.fm.

| | Model | Yelp RMSE (SD) | Yelp MAE (SD) | Last.fm RMSE (SD) | Last.fm MAE (SD) |
|---|---|---|---|---|---|
| (a) Mean-centering | User average rating | 2.313 (0.008) | 1.656 (0.008) | 1.043 (0.004) | 0.873 (0.004) |
| | Item average rating | 1.215 (0.003) | 0.932 (0.001) | 1.399 (0.009) | 1.092 (0.008) |
| | Overall average rating | 1.280 (0.005) | 1.030 (0.004) | 1.792 (0.004) | 1.464 (0.004) |
| | HyPER (all mean-centering rules) | **1.199** (0.003) | 0.952 (0.002) | **1.032** (0.004) | **0.861** (0.004) |
| (b) User-based | Similar users (Pearson) | 2.313 (0.008) | 1.656 (0.008) | 1.043 (0.004) | 0.874 (0.004) |
| | Similar users (cosine) | 2.313 (0.008) | 1.657 (0.008) | 1.043 (0.004) | 0.873 (0.004) |
| | Similar users (latent, cosine) | 2.227 (0.007) | 1.597 (0.007) | 1.025 (0.004) | 0.862 (0.004) |
| | Similar users (latent, Euclidean) | 2.226 (0.009) | 1.596 (0.008) | 1.025 (0.004) | 0.863 (0.004) |
| | HyPER (all user-based rules) | **2.194** (0.008) | **1.573** (0.008) | 1.025 (0.004) | **0.861** (0.004) |
| (c) Item-based | Similar items (Pearson) | 1.213 (0.004) | 0.931 (0.002) | 1.397 (0.008) | 1.098 (0.006) |
| | Similar items (cosine) | 1.211 (0.003) | 0.928 (0.001) | 1.396 (0.008) | 1.100 (0.007) |
| | Similar items (adjusted cosine) | 1.210 (0.004) | 0.924 (0.002) | 1.405 (0.008) | 1.092 (0.007) |
| | Similar items (latent, cosine) | 1.212 (0.003) | 0.923 (0.001) | 1.379 (0.009) | 1.080 (0.008) |
| | Similar items (latent, Euclidean) | 1.212 (0.003) | 0.931 (0.001) | 1.379 (0.008) | 1.081 (0.008) |
| | HyPER (all item-based rules) | **1.208** (0.004) | 0.923 (0.002) | **1.362** (0.007) | **1.070** (0.006) |
| (d) Content &Social | Similar items (content) | 1.200 (0.003) | 0.939 (0.002) | 1.029 (0.004) | 0.867 (0.004) |
| | Friends | 1.199 (0.003) | 0.932 (0.002) | 1.013 (0.004) | 0.853 (0.004) |
| | HyPER (content + social rules) | **1.195** (0.003) | **0.927** (0.002) | 1.013 (0.004) | **0.857** (0.004) |
| (e) Base models | Item-based | 1.216 (0.004) | 0.932 (0.001) | 1.408 (0.010) | 1.096 (0.008) |
| | MF | 1.251 (0.006) | 0.944 (0.005) | 1.178 (0.003) | 0.939 (0.003) |
| | BPMF | 1.191 (0.003) | 0.954 (0.003) | 1.008 (0.005) | 0.839 (0.004) |
| | HyPER (baseline rules) | **1.179** (0.003) | **0.926** (0.002) | **1.005** (0.005) | **0.836** (0.004) |
| | HyPER (all rules) | **1.173** (0.003) | **0.917** (0.002) | **1.001** (0.004) | **0.833** (0.004) |

bines the results of those recommenders, without any additional rules. The combined model performed better than the individual baselines.

**Relative Importance of Information Sources:** When performing weight learning (Section 3.3), the learned weights of the rules are indicative of the relative importance of the signals. For Last.fm, average user ratings had a high rule weight while average item ratings did not, while the reverse was true for Yelp, suggesting a difference in the importance of user judiciousness versus item popularity between the data sets. Item similarities had a high weight for Last.fm, while MF predictions had a high weight for Yelp. Negated rules, which decrease predicted ratings, were typically weighted lower than their non-negated counterparts. In general, the rules for BPMF predictions had high weights.

## 5. RELATED WORK

There is a large body of work on recommender systems; see Ricci et al. [16] for an overview. We focus our related work discussion on hybrid recommender systems, and particularly systems that can incorporate multi-relational and heterogeneous data as well as graphical modeling approaches. In Burke [17]'s taxonomy of hybrid recommender systems our work falls into the "feature augmentation" category.

Hybrid systems typically combine two or more approaches in order to provide better recommendations, usually content-based and collaborative filtering approaches [18, 19] or variations of collaborative filtering approaches [20]. Gunawardana and Meek [18] present a domain-agnostic hybrid approach for using content to improve item-item modeling. After the Netflix Prize competition, ensemble methods [21] have gained popularity. Factorization Machines [22] are a general matrix factorization method that can be applied to design hybrid factorization models. Recently, as user-generated content has become available, researchers have studied how to leverage information such as social relationships [5, 6], reviews [23, 24], tags [25], and feedback [26] to improve recommendations. Incorporating additional information for users and/or items is especially beneficial in cold-start settings [27]. Dooms [28] argues that a flexible recommendation system that automatically generates good hybrid models would be very valuable as information sources increase. Our model provides such flexibility, allowing for the combination of as many information sources as are available. Fakhraei et al. [13] use PSL to reason over multiple similarity measures for predicting drug-target interactions. Our approach extends this model in several important ways for the recommender systems domain.

Chen at al. [29] learn the strength of ties between users based on multi-relational network information. The learned network is combined with item-based collaborative filtering to improve recommendation results. Burke et al. [30, 31] integrate different dimensions of data about users in a heterogenous network by using metapaths to create multiple two-dimensional projections representing relationships between entities (e.g. users-tags) and then linearly combining these projections. Also using metapaths, Yu et al. [32] propose a global and a personalized recommendation model. In their approach, implicit feedback is incorporated into metapaths and latent features for users and items are generated using matrix factorization.

De Campos et al. [3] propose a probabilistic graphical modeling recommendation approach using Bayesian networks. Their approach combines individual predictions from content-

based and user-based collaborative filtering components. Hoxha and Rettinger [33] also discuss a probabilistic graphical modeling representation, using Markov Logic Networks (MLNs) [34] to combine content with collaborative filtering. Both MLNs and HL-MRFs operate on undirected graphical models using a first-order logic as their template language, while Bayesian networks are directed. We chose HL-MRFs because they can represent ordered data such as ratings, and due to their scalability with parallel convex optimization for inference. Speed and scalability is of paramount importance in recommender systems and in particular when we run the prediction task collectively over multiple types of input data with a variety of similarity measures.

# 6. CONCLUSION

In this paper we presented HyPER, a new hybrid recommender system which is flexible, problem-agnostic, and is easily extensible via a probabilistic programming interface. HyPER uses a hinge-loss MRF formulation, allowing scalable and accurate inference. Our comprehensive experiments demonstrate that HyPER can learn to appropriately balance many information sources, resulting in improved performance over previous state-of-the-art approaches on two benchmark datasets.

## Acknowledgements

We would like to thank Ben London and Alex Ntoulas for insightful discussions. We thank Juntao Liu for sharing the code of their approach [5], George Karypis, Christian Desrosiers, Chris Meek, Asela Gunawardana, and Robin Burke for their help. This work was partially supported by NSF grant IIS1218488 and by the IARPA via DoI/NBC contract D12PC00337. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, IARPA, DoI/NBC, or the U.S. Government.

# 7. REFERENCES

[1] S.H. Bach, M. Broecheler, B. Huang, and L. Getoor. Hinge-loss markov random fields and probabilistic soft logic. *ArXiv:1505.04406 [cs.LG]*, 2015.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Transactions on Knowledge and Data Engineering*, 17(6), 2005.

[3] L. de Campos, J. Fernández-Luna, J. Huete, and M. Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks. *International Journal of Approximate Reasoning*, 51(7), 2010.

[4] C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender Systems Handbook*. Springer, 2011.

[5] J. Liu, C. Wu, and W. Liu. Bayesian probabilistic matrix factorization with social relations and item contents for recommendation. *Decision Support Systems*, 55(3), 2013.

[6] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *WSDM*, 2011.

[7] N. Goodman, V. Mansinghka, D.M. Roy, K. Bonawitz, and J. Tenenbaum. Church: a language for generative models with non-parametric memoization and approximate inference. In *UAI*, 2008.

[8] L. Getoor and B. Taskar. *Introduction to statistical relational learning*. MIT press, 2007.

[9] S. H. Bach, B. Huang, B. London, and L. Getoor. Hinge-loss Markov random fields: Convex inference for structured prediction. In *UAI*, 2013.

[10] M. Broecheler, L. Mihalkova, and L. Getoor. Probabilistic similarity logic. In *UAI*, 2010.

[11] P. Lops, M. Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*. Springer, 2011.

[12] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8), 2009.

[13] S. Fakhraei, B. Huang, L. Raschid, and L. Getoor. Network-based drug-target interaction prediction with probabilistic soft logic. *Transactions on Computational Biology and Bioinformatics*, 11(5), 2014.

[14] P. D. Hoff, A. E. Raftery, and M. S. Handcock. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97, 2001.

[15] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *ICML*, 2008.

[16] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. *Recommender Systems Handbook*. Springer, 2011.

[17] R. Burke. Hybrid web recommender systems. In *The Adaptive Web*. Springer, 2007.

[18] A. Gunawardana and C. Meek. A unified approach to building hybrid recommender systems. In *RecSys*, 2009.

[19] P. Forbes and M. Zhu. Content-boosted matrix factorization for recommender systems: Experiments with recipe recommendation. In *RecSys*, 2011.

[20] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *KDD*, 2008.

[21] M. Jahrer, A. Töscher, and R. Legenstein. Combining predictions for accurate recommender systems. In *KDD*, 2010.

[22] S. Rendle. Factorization machines with libFM. *ACM Transactions on Intelligent Systems and Technology*, 3(3), 2012.

[23] J. McAuley and J. Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *RecSys*, 2013.

[24] G. Ling, M. R. Lyu, and I. King. Ratings meet reviews, a combined approach to recommend. In *RecSys*, 2014.

[25] I. Guy, N. Zwerdling, I. Ronen, D. Carmel, and E. Uziel. Social media recommendation based on people and tags. In *SIGIR*, 2010.

[26] S. Sedhain, S. Sanner, D. Braziunas, L. Xie, and J. Christensen. Social collaborative filtering for cold-start recommendations. In *RecSys*, 2014.

[27] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *ICDM*, 2010.

[28] S. Dooms. Dynamic generation of personalized hybrid recommender systems. In *RecSys*, 2013.

[29] J. Chen, G. Chen, H. Zhang, J. Huang, and G. Zhao. Social recommendation based on multi-relational analysis. In *WI-IAT*, 2012.

[30] R. Burke, F. Vahedian, and B. Mobasher. Hybrid recommendation in heterogeneous networks. In *User Modeling, Adaptation, and Personalization*. Springer, 2014.

[31] J. Gemmell, T. S., B. Mobasher, and R. Burke. Resource recommendation in social annotation systems: A linear-weighted hybrid approach. *Journal of Computer and System Sciences*, 78(4), 2012.

[32] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han. Personalized entity recommendation: A heterogeneous information network approach. In *WSDM*, 2014.

[33] J. Hoxha and A. Rettinger. First-order probabilistic model for hybrid recommendations. In *ICMLA*, 2013.

[34] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2), 2006.