Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks

Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song and Dik Lun Lee
{hzhaoaf,qyaoaa}@cse.ust.hk;jlicy@connect.ust.hk;{yqsong,dlee}@cse.ust.hk
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong

ABSTRACT

Heterogeneous Information Network (HIN) is a natural and general representation of data in modern large commercial recommender systems which involve heterogeneous types of data. HIN based recommenders face two problems: how to represent the high-level semantics of recommendations and how to fuse the heterogeneous information to make recommendations. In this paper, we solve the two problems by first introducing the concept of meta-graph to HIN-based recommendation, and then solving the information fusion problem with a "matrix factorization (MF) + factorization machine (FM)" approach. For the similarities generated by each meta-graph, we perform standard MF to generate latent features for both users and items. With different meta-graph based features, we propose to use FM with Group lasso (FMG) to automatically learn from the observed ratings to effectively select useful meta-graph based features. Experimental results on two real-world datasets, Amazon and Yelp, show the effectiveness of our approach compared to state-of-the-art FM and other HIN-based recommendation algorithms.

CCS CONCEPTS

•Information systems → Collaborative filtering; Recommender systems; •Computer systems organization → *Heterogeneous (hybrid) systems;*

KEYWORDS

Recommendation system; Collaborative filtering; Heterogeneous information networks; Factorization machine.

1 INTRODUCTION

Recommendation on the platforms like Amazon or Yelp refers to the problem of recommending items, such as products or businesses, to users so that the platforms can make more revenue when users consume more items. Essentially if we consider users and items as a bipartite graph, this is a link prediction problem on heterogeneous types of entities, i.e., *User* and *Item*. Nowadays large commercial recommender systems often incorporate richer heterogeneous information. For example, on Amazon, the products have categories, or they can belong to brands, and users can write reviews to products. On Yelp, users can follow other users to form a social network, the location based businesses have categories, and users can write reviews to businesses as well. Then, real-world recommender systems often need to consider richer semantics with different types of information that are enabled and collected by the platforms. This richer heterogeneity thus requires the development of a mathematical representation to formulate it and a tool to compute over it.

Heterogeneous information networks (HINs) [30] have been proposed as a general data representation for many different types of data, such as scholar network data [32], social network data [13], patient network data [3], or knowledge graph data [5]. At the beginning, HINs were used to handle entity search and similarity measure problems [32], where the query and result entities are assumed to be of the same type (e.g., using *Person* to search *Person*). Later, it was extended to handle heterogeneous entity recommendation problems (i.e., recommending *Items* to *Users*) [29, 39, 40]. To incorporate rich semantics, HINs first builds a network schema of the heterogeneous network. For example, for Yelp, a network schema is defined over the entity types *User*, *Review*, *Word*, *Business*, etc. Then, the semantic relatedness constrained by the entity types can be defined by the similarities between two entities along meta-paths [32]. For traditional collaborative filtering, if we want to recommend businesses to users, we can build a simple meta-path *Business*→*User* and learn from this meta-path to make generalizations. From HIN's schema, we can define more complicated meta-paths like *User* → *Review* → *Word* → *Review* → *Business*. This meta-path defines a similarity to measure whether a user tends to like a business if his/her reviews are similar to those written by other users for the same business.

When applying meta-path based similarities to recommender systems, there are two major challenges. First, meta-path may not be the best way to characterize the rich semantics. Figure 1 shows a concrete example, where a meta-path *User* → *Review* → *Word* → *Review* → *Business* is used to capture users' similarity since they both write reviews and mention the seafood it serves. However, if we want to capture the semantic that U_1 and U_2 rate the same type of business (such as *Restaurant*), and at the same time, they mention the same aspect (such as *seafood*), the meta-path fails. Thus, we need a better way to capture such complicated semantics. Recently, Fang et al. [6] and Huang et al. [10] have proposed to use meta-graph (or meta-structure) to compute similarity between homogeneous type of entities (e.g., using *Person* to search *Person*) over HINs, which can capture more complex semantics that meta-path cannot. However, they didn't explore entities of heterogeneous

KDD'17, August 13–17, 2017, Halifax, NS, Canada.

DOI: http://dx.doi.org/10.1145/3097983.3098063

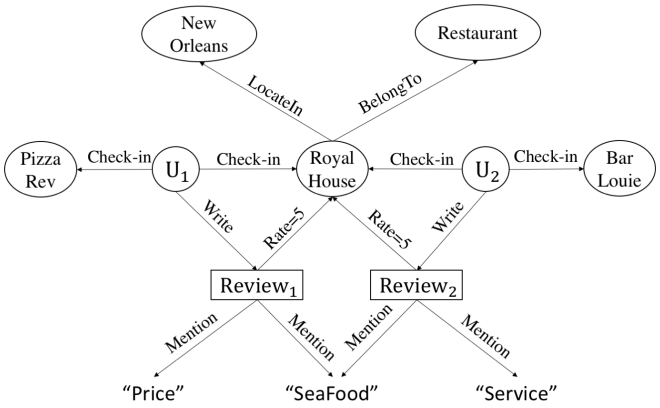


Figure 1: Example of HIN, which is built based on the web page for Royal House on Yelp.

types with meta-graph. Thus in this paper, we extend this idea to heterogeneous recommendation problem. However, how to use the similarities between heterogeneous types of entities developed from HINs in recommendation is still unclear, which results in the second challenge.

Second, different meta-paths or meta-graphs result in different similarities. How to assemble them in an effective way is another challenge. Currently, there are two principled ways. Considering our goal is to achieve accurate prediction of *User* and *Item* ratings, which can be formulated as a matrix completion problem of the user-item rating matrix. One way to predict the missing ratings based on HIN is to use meta-paths to generate a lot of ad-hoc alternative similarities for user-item matrix, and then learn a weighting mechanism for different meta-paths to combine the similarities explicitly to approximate the user-item rating matrix [29]. This approach does not consider implicit factors of each meta-path, and each alternative similarity matrix could be very sparse to contribute to the final ensemble. The other way is to first factorize each user-item similarity matrix computed based on each meta-path, and then use the latent features to recover a new user-item matrix, which is used for ensemble [40]. This method resolves the sparsity problem of each similarity matrix. However, it does not fully make use of the latent features since when ensemble is performed, each meta-path cannot see others' variables but only the single value predicted by the others.

To address the above challenges, we propose a new principled way to fully combine different latent features. First, instead of using meta-paths for heterogeneous recommendation [29, 40], we introduce the concept of meta-graph to the recommendation problem, which allows us to incorporate more complex semantics into our prediction problem. Second, instead of computing the recovered matrices directly, we use all of the latent features of all meta-graphs. Inspired by the famous work PCA+LDA used for face recognition [2], which first uses PCA (principle component analysis) to perform unsupervised dimensionality reduction, and then applies LDA (linear discriminant analysis) to discover further reduced dimensions guided by supervision, we apply matrix factorization

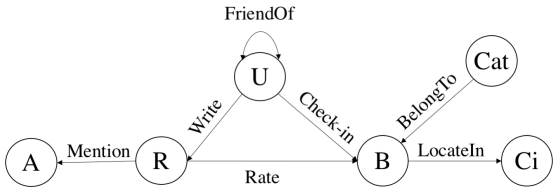


Figure 2: Example of HIN Schema. A: aspect extracted from reviews; R: reviews; U: users; B: business; Cat: category of item; Ci: city.

(MF) + factorization machine (FM) [25] to our recommendation problem. For each meta-graph, we first compute the user-item similarity matrix under the guidance of the meta-graph, and then use unsupervised (without seeing the ratings) MF to factorize it into a set of user and item latent vectors. Then, with many different sets of user and item vectors computed from different meta-graphs, we use FM to assemble them to learn from the rating matrix. To effectively select useful meta-graphs, we propose to use FM with Group lasso (FMG), i.e. the $\ell_{2,1}$-norm regularization, to learn the parameters. In this way, we can automatically determine for new incoming problems which meta-graph should be used, and for each meta-graph generated user and item vectors, how they should be weighted. Experimental results on two large real-world datasets, Amazon and Yelp, show that our approach can successfully outperform other MF-based, FM-based, and existing HIN-based state-of-the-arts for recommendation. Our code is available at https://github.com/HKUST-KnowComp/FMG.

2 FRAMEWORK

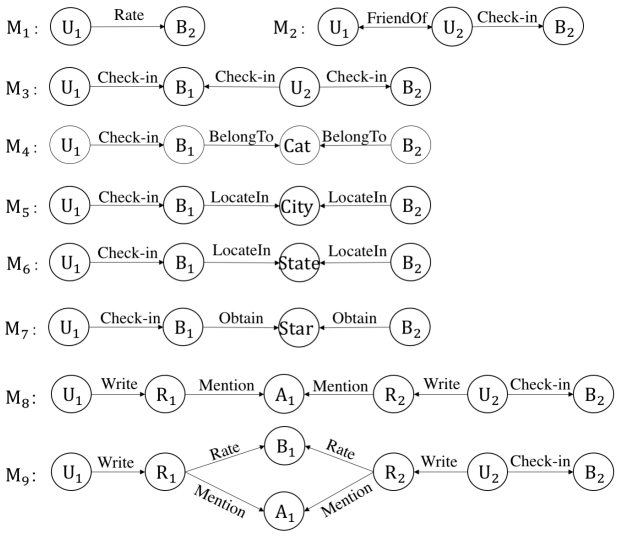
In this section, we introduce our framework to handle HIN-based recommendation.

2.1 Meta-graph based Similarity

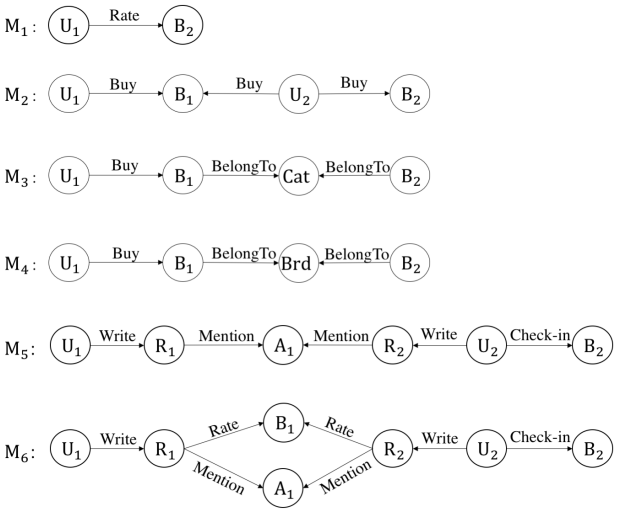
The definitions of HIN and HIN Schema (a schema graph of entity types and their relations) have been introduced in [32]. Here we skip the formal definition and only illustrate the original HIN in Figure 1 and the corresponding schema in Figure 2. Here we focus on the concepts related to our paper. First, we formally define the meta-graph in HIN for recommendation.

Definition 2.1. **Meta-graph.** A meta-graph $\mathcal{M}$ is a directed acyclic graph (DAG) with a single source node n_s (i.e., with in-degree 0) and a single sink (target) node n_t (i.e., with out-degree 0), defined on an HIN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with schema $\mathcal{T}_G = (\mathcal{A}, \mathcal{R})$, where $\mathcal{V}$ is the node set, $\mathcal{E}$ is the edge set, $\mathcal{A}$ is the node type set, and $\mathcal{R}$ is the edge type set. Then we define a meta-graph as $\mathcal{M} = (\mathcal{V}_M, \mathcal{E}_M, \mathcal{A}_M, \mathcal{R}_M, n_s, n_t)$, where $\mathcal{V}_M \subseteq \mathcal{V}$, $\mathcal{E}_M \subseteq \mathcal{E}$ constrained by $\mathcal{A}_M \subseteq \mathcal{A}$ and $\mathcal{R}_M \subseteq \mathcal{R}$, respectively.

We show all of the meta-graphs used in this paper for both Amazon and Yelp in Figure 3. We can see that they are DAGs with U (*User*) as the source node and B (*Business* for Yelp and *Product* for Amazon) as the target node. Here we use $\mathcal{M}_3$ and $\mathcal{M}_9$ used on Yelp data to illustrate the computation problem.



(a) Yelp-200K (Star: the average stars a business obtained).



(b) Amazon-200K (Brd: brand of the item).

Figure 3: Meta-graphs used for Amazon and Yelp datasets.

Given the above definition of meta-graph, we want to compute the similarities between the source and the target nodes. Originally, commuting matrices [32] have been used to compute the counting-based similarity matrix of a meta-path. Suppose we have a meta-path $\mathcal{P} = (A_1, A_2, \ldots, A_l)$, where A_i's are node types in $\mathcal{A}$. Then we can define a matrix $\mathbf{W}_{A_iA_j}$ as the adjacency matrix between type A_i and type A_j. Then, the commuting matrix for path $\mathcal{P}$ is $\mathbf{C}_P = \mathbf{W}_{A_1,A_2} \cdot \mathbf{W}_{A_2,A_3} \cdot \ldots \cdot \mathbf{W}_{A_{l-1},A_l}$. For example, for $\mathcal{M}_3$ in Figure 3(a), $\mathbf{C}_{M_3} = \mathbf{W}_{UB} \cdot \mathbf{W}_{UB}^\top \cdot \mathbf{W}_{UB}$, where $\mathbf{W}_{UB}$ is the adjacency matrix between type U and type B. This shows that the counting-based similarities for a meta-path can be computed by the multiplication of a sequences of matrices like the above $\mathbf{W}_{UB}$. In practice, we can implement this in a very efficient way if the adjacency matrices $\mathbf{W}$'s are sparse.

For meta-graphs, the problem becomes more complicated. For example, for $\mathcal{M}_9$ in Figure 3(a), there are two ways to pass through the meta-graph, which are (U, R, A, R, U, B) and (U, R, B, R, U, B). Note that R represents the entity type *Review* in HIN. Here in the path (U, R, A, R, U, B), (R, A, R) means that if two reviews both mention the same A (*Aspect*), then they have some similarity. Similarly, in (U, R, B, R, U, B), R, B, R means that if two reviews both rate the same B (*Business*), then they have some similarity as well. We should define our logic of similarity when there are multiple ways for a flow passing through the meta-graph from source node to the target one. When there are two paths, we can allow a flow to pass through either path, or we constrain a flow to satisfy both of them. By analyzing the former strategy, we find that it is similar to simply split such meta-graph into multiple meta-paths and then adopt our later computation. Thus, we choose the latter, which requires one more matrix operation than simple multiplication, i.e., the Hadamard product, or element-wise product. Algorithm 1 depicts the algorithm for computing the counting-based similarity for $\mathcal{M}_9$ in Figure 3(a), where $\odot$ is the Hadamard product. After obtaining $\mathbf{C}_{S_r}$, it is easier to obtain the whole commuting matrix $\mathbf{C}_{M_9}$ by the multiplication of a sequence of matrices. In practice, not limited to $\mathcal{M}_9$ in Figure 3(a), the meta-graph defined in this paper can be computed by two operations (Hadamard product and multiplication) on the corresponding matrices.

Algorithm 1 Computing commuting matrix for $\mathbf{C}_{M_9}$.

1: Compute $\mathbf{C}_{P_1} : \mathbf{C}_{P_1} = \mathbf{W}_{RB} \cdot \mathbf{W}_{RB}^\top$;
2: Compute $\mathbf{C}_{P_2} : \mathbf{C}_{P_2} = \mathbf{W}_{RA} \cdot \mathbf{W}_{RA}^\top$;
3: Compute $\mathbf{C}_{S_r} : \mathbf{C}_{S_r} = \mathbf{C}_{P_1} \odot \mathbf{C}_{P_2}$;
4: Compute $\mathbf{C}_{M_9} : \mathbf{C}_{M_9} = \mathbf{W}_{UR} \cdot \mathbf{C}_{S_r} \cdot \mathbf{W}_{UR}^\top \cdot \mathbf{W}_{UB}$.

By computing the similarities between all users and items along the meta-graph $\mathcal{M}$, we can obtain a user-item similarity matrix $\hat{\mathbf{R}} \in \mathbb{R}^{m \times n}$, where $\hat{\mathbf{R}}_{ij}$ represents the similarity between user u_i and item b_j along the meta-graph $\mathcal{M}$, and m and n are the number of users and items, respectively. Then by designing L meta-graphs, we can get L different user-item similarity matrices, denoted by $\hat{\mathbf{R}}^1, \ldots, \hat{\mathbf{R}}^L$.

2.2 Meta-graph based Latent Features

After we obtain L different user-item similarity matrices, we use matrix factorization to obtain the latent features of users and items to reduce the noise and fix the sparsity problem of the original similarity matrices. State-of-the-art MF techniques can be used for the task [14, 22, 36]. Based on the assumption that the users' preferences are controlled by a small number of factors, the similarity matrix $\mathbf{R}$ can be factorized into two low-rank matrices, $\mathbf{U}$ and $\mathbf{B}$, which represent the latent features of users' preferences and

items, respectively. By solving the optimizing problem in (1), the low dimensional representations of users and items can be obtained:

$$\min_{\mathbf{U},\mathbf{B}} \frac{1}{2}||P_\Omega(\mathbf{U}\mathbf{B}^\top - \mathbf{R})||_F^2 + \frac{\lambda_u}{2}||\mathbf{U}||_F^2 + \frac{\lambda_b}{2}||\mathbf{B}||_F^2, \tag{1}$$

where observed positions are indicated by 1's in $\Omega \in \{0,1\}^{u_m \times b_n}$, and $[P_\Omega(\mathbf{X})]_{ij} = \mathbf{X}_{ij}$ if $\Omega_{ij} = 1$, and 0 otherwise. λ_u and λ_b are the hyper-parameters that control the influence of Frobenius norm regularization to avoid overfitting. For L meta-graph based similarities between users and items, we can obtain L groups of latent features of users and items, denoted as $\mathbf{U}^{(1)}, \mathbf{B}^{(1)}, ..., \mathbf{U}^{(L)}, \mathbf{B}^{(L)}$.

2.3 Recommendation Model

After we obtain L groups of user and item latent features, for a sample $\mathbf{x}^n$ in the observed ratings, i.e., a pair of user and item, denoted by $\mathbf{u}_i$ and $\mathbf{b}_j$, we concatenate all of the corresponding user and item features from all of the L meta-graphs:

$$\mathbf{x}^n = \underbrace{\mathbf{u}_i^{(1)}, ..., \mathbf{u}_i^{(l)}, ..., \mathbf{u}_i^{(L)}}_{L\times F} \underbrace{\mathbf{b}_j^{(1)}, ..., \mathbf{b}_j^{(l)}, ..., \mathbf{b}_j^{(L)}}_{L\times F}, \tag{2}$$

where F is the rank used to factorize every similarity matrix by (1), and $\mathbf{u}_i^{(l)}$ and $\mathbf{b}_j^{(l)}$, respectively, represent user and item latent features generated from l-th meta-graph. Note that F can be different for different matrices, but we keep it constant for simplicity. $\mathbf{x}^n$ represents the feature vector of the n-th sample after concatenation. Then each user and item can be represented by the $L \times F$ latent features, respectively.

Given all of the features in (2), the rating for the sample $\mathbf{x}^n$ based on FM [25] is computed as follows:

$$\hat{y}^n(\mathbf{w},\mathbf{V}) = w_0 + \sum_{i=1}^{d} w_i x_i^n + \sum_{i=1}^{d}\sum_{j=i+1}^{d} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i^n x_j^n, \tag{3}$$

where w_0 is the global bias, $\mathbf{w} \in \mathbb{R}^d$, representing the first-order weights for the features, and $\mathbf{V} = [\mathbf{v}_i] \in \mathbb{R}^{d\times K}$ represents the second-order weights to model the interactions across different features. $\langle \cdot, \cdot \rangle$ is the dot product of two vectors of size K. $\mathbf{v}_i$ is the i-th row of the matrix $\mathbf{V}$, which describes the i-th variable with K factors. $d = 2LF$ represents the total number of features generated by the L meta-graph based similarity matrices. F is the rank used to factorize every similarity matrix. $\mathbf{x}_i^n$ is the i-th feature in $\mathbf{x}^n$. The parameters can be learned by minimizing the mean square loss:

$$\min_{\mathbf{w},\mathbf{V}} \sum_{n=1}^{N} (y^n - \hat{y}^n(\mathbf{w},\mathbf{V}))^2, \tag{4}$$

where y^n is an observed rating for the n-th sample. N is the number of all the observed ratings.

There are two problems when applying the FM model to the meta-graph based latent features. The first problem is that it may bring noise when working with many meta-graphs thus impairing the predicting capability of the model. Moreover, in practice, some meta-graphs can be useless since information provided by some meta-paths can be covered by others. The second problem is the computational cost. All of the features are generated by standard matrix factorization, which means that the design matrix, i.e., features fed to FM, are dense. It increases the computational cost for learning the parameters of the model as well as that of online recommendation.

To alleviate the above two problems, we propose a novel regularization for FM, i.e., the group lasso regularization [42], which is a feature selection method on a group of variables. The group lasso regularization of parameters $\mathbf{p}$ is defined as follows:

$$\Phi(\mathbf{p}) = \sum_{g=1}^{G} ||\mathbf{p}_{\mathcal{I}_g}||_2, \tag{5}$$

where $\mathcal{I}_g$ is the index set belonging to the predefined g-th group of variables, $g = 1, 2, ..., G$, and $||\cdot||_2$ is the ℓ_2-norm. In our model, the groups correspond to the meta-graph based features. For example, $\mathbf{U}^{(l)}$ and $\mathbf{B}^{(l)}$ are the user and item latent features generated by the l-th meta-graph. For a pair of user i and item j, the latent features are $\mathbf{u}_i^{(l)}$ and $\mathbf{b}_i^{(l)}$. There are two corresponding groups of variables in $\mathbf{w}$ and $\mathbf{V}$ according to (3). With L meta-graphs, the features of users and items from every single meta-graph can be put in a group. We have in total $2L$ groups of variables in $\mathbf{w}$ and $\mathbf{V}$, respectively.

For the first-order parameters $\mathbf{w}$ in (3), which is a vector, the group lasso is applied to the subset of variables in $\mathbf{w}$. Then we have:

$$\Phi_{\mathbf{w}}(\mathbf{w}) = \sum_{l=1}^{2L} ||\mathbf{w}_l||_2, \tag{6}$$

where $\mathbf{w}_l \in \mathbb{R}^F$, which models the weights for a group of user or item features from one meta-graph. For the second-order parameters $\mathbf{V}$ in (3), we have the regularizer as:

$$\Phi_{\mathbf{V}}(\mathbf{V}) = \sum_{l=1}^{2L} ||\mathbf{V}_l||_F, \tag{7}$$

where $\mathbf{V}_l \in \mathbb{R}^{F\times K}$, the l-th block of $\mathbf{V}$ corresponding to the l-th meta-graph based features in a sample, and $||\cdot||_F$ is the Frobenius norm.

With group lasso regularizations, during the training process, our model can automatically select useful features and remove redundant ones in group, i.e. generated by different meta-graphs. In the selection process, unuseful features are removed in the unit of a group.

2.4 Comparison with Previous Latent Feature based Model

Previous approaches of recommendation based on HIN [40] also applied matrix factorization to generate latent features from different meta-paths and predict the rating by a weighted ensemble of dot product of user and item latent features from every single meta-path:

$$\hat{r}(\mathbf{u}_i, \mathbf{b}_j) = \sum_{l=1}^{L} \theta_l \cdot \hat{\mathbf{u}}_i^{(l)} \cdot \hat{\mathbf{b}}_j^{(l)T}, \tag{8}$$

where $\hat{r}(\mathbf{u}_i, \mathbf{b}_j)$ is the predicted rating for user $\mathbf{u}_i$ and $\mathbf{b}_j$, L is the number of meta-paths used, and θ_l is the weight for the l-th meta-path latent features. However, here the predicting method is not adequate, as it fails to capture the interactions between inter-meta-graph features, i.e. features across different meta-graphs, as well

as the intra-meta-graph features, i.e. features inside a single meta-graph. It may decrease the prediction ability of all of the user and item features.

3 MODEL OPTIMIZATION

In this section, we introduce how to solve the optimization problem. We define our FM over meta-graph (FMG) model with the following objective function:

$$h(\mathbf{w},\mathbf{V}) = \sum_{n=1}^{N} (y^n - \hat{y}^n(\mathbf{w},\mathbf{V}))^2 + \lambda_w \Phi_{\mathbf{w}}(\mathbf{w}) + \lambda_v \Phi_{\mathbf{V}}(\mathbf{V}). \quad (9)$$

Note that in (3) we merge all of the superscripts (l)'s into subscript without introducing confusion of the original FM model. Here, we can see that h is non-smooth due to the use of $\Phi_{\mathbf{w}}$ and $\Phi_{\mathbf{V}}$. Besides, as (3) is not convex on $\mathbf{V}$, h is also not convex.

3.1 Optimization

To tackle the non-convex non-smooth objective function, we propose to use the proximal gradient algorithm [23], which is a powerful tool to handle non-convex problems, in the form of (9). Specifically, the state-of-the-art nonmonotonous accelerated proximal gradient (nmAPG) algorithm [17] is used. The motivation comes from two facts. First, nonsmoothness comes from the proposed regularizers, which can be efficiently handled since the corresponding proximal steps have cheap closed-form solution. Second, the acceleration technique is useful for significantly speeding up first order optimization algorithms [17, 37], and nmAPG is the only technique which can deal with general non-convex problems with sound convergence guarantee. The whole procedure is given in Algorithm 2.

Note that while both $\Phi_{\mathbf{w}}$ and $\Phi_{\mathbf{V}}$ are nonsmooth in (9), they are imposed on $\mathbf{w}$ and $\mathbf{V}$ separately. Thus, we can also perform proximal step independently for these two regularizers [23] as follows.

$$\mathbf{prox}_{\lambda_w\Phi_{\mathbf{w}}+\lambda_V\Phi_{\mathbf{V}}}(\mathbf{w},\mathbf{V}) = \left(\mathbf{prox}_{\lambda_w\Phi_{\mathbf{w}}}(\mathbf{w}), \mathbf{prox}_{\lambda_V\Phi_{\mathbf{V}}}(\mathbf{V})\right).$$

These are performed at lines 6-7 and 12-13. The closed-form solution of $\mathbf{prox}_{\alpha\Phi_{\mathbf{w}}}(\cdot)$ and $\mathbf{prox}_{\alpha\Phi_{\mathbf{V}}}(\cdot)$ can be obtained easily from Lemma 3.1 below. Thus, each proximal step can be solved in one pass of all groups.

Lemma 3.1 ([41]). *The closed-form solution of* $\mathbf{p}^* = \mathbf{prox}_{\lambda\Phi}(z)$ *(Φ is defined in* (5)*) is given by*

$$\mathbf{p}^*_{\mathcal{I}_g} = \max\left(1 - \frac{\lambda}{\|\mathbf{z}_{\mathcal{I}_g}\|_2}, 0\right)\mathbf{z}_{\mathcal{I}_g},$$

for all $g = 1, \ldots, G$.

Finally, it is easy to verify that the assumptions on the convergence of nmAPG in [17] are satisfied. Thus, Algorithm 2 is guaranteed to produce a critical point of (9).

3.2 Complexity Analysis

The major computational cost in the training process is to update the gradients of all parameters including gradient calculation and evaluation of proximal operators. In [25], the author shows that for every single sample, computing each gradient is $O(1)$ time, which leads to $O(Kd)$ time in total for one sample, where K is the dimensions for factorizing the second order parameters, as shown in (3), and $d = 2LF$ is the number of features in every sample. For evaluating the proximal operator Φ, according to Lemma 3.1, for each sample, each gradient also costs $O(1)$, thus achieving $O(Kd)$ time by updating all gradients in one sample. Then, the total time in one iteration is $O(N(Kd + Kd)) = O(NKd)$, where N is the number of all observations. Assuming T iterations are used in total, the overall time of the learning process is $O(TNKd)$.

Algorithm 2 nmAPG [17] algorithm for (9).

1: Initiate $\mathbf{w}_0$, $\mathbf{V}_0$ as Gaussian random matrices;
2: $\bar{\mathbf{w}}_1 = \mathbf{w}_1 = \mathbf{w}_0$, $\bar{\mathbf{V}}_1 = \mathbf{V}_1 = \mathbf{V}_0$, $c_1 = h(\mathbf{w}_1, \mathbf{V}_1)$; $q_1 = 1$, $\delta = 10^{-3}$, $a_0 = 0$, $a_1 = 1$, $\alpha = 10^{-7}$;
3: **for** $t = 1, 2, 3, \ldots, T$ **do**
4: $\quad \mathbf{y}_t = \mathbf{w}_t + \frac{a_{t-1}}{a_t}(\bar{\mathbf{w}}_t - \mathbf{w}_t) + \frac{a_{t-1}-1}{a_t}(\mathbf{w}_t - \mathbf{w}_{t-1})$;
5: $\quad \mathbf{Y}_t = \mathbf{V}_t + \frac{a_{t-1}}{a_t}(\bar{\mathbf{V}}_t - \mathbf{V}_t) + \frac{a_{t-1}-1}{a_t}(\mathbf{V}_t - \mathbf{V}_{t-1})$;
6: $\quad \bar{\mathbf{w}}_{t+1} = \mathbf{prox}_{\alpha\lambda\Phi_{\mathbf{w}}}(\mathbf{w}_t - \alpha\nabla_{\mathbf{w}} h(\mathbf{w}_t, \mathbf{V}_t))$;
7: $\quad \bar{\mathbf{V}}_{t+1} = \mathbf{prox}_{\alpha\lambda\Phi_{\mathbf{V}}}(\mathbf{V}_t - \alpha\nabla_{\mathbf{V}} h(\mathbf{w}_t, \mathbf{V}_t))$;
8: $\quad \Delta_t = \|\bar{\mathbf{w}}_{t+1} - \mathbf{y}_t\|_2^2 + \|\bar{\mathbf{V}}_{t+1} - \mathbf{Y}_t\|_F^2$
9: $\quad$ **if** $h(\bar{\mathbf{w}}_{t+1}, \bar{\mathbf{V}}_{t+1}) \le c_t - \delta\Delta_t$; **then**
10: $\quad\quad \mathbf{w}_{t+1} = \bar{\mathbf{w}}_{t+1}, \mathbf{V}_{t+1} = \bar{\mathbf{V}}_{t+1}$;
11: $\quad$ **else**
12: $\quad\quad \hat{\mathbf{w}}_{t+1} = \mathbf{prox}_{\alpha\lambda\Phi_{\mathbf{w}}}(\mathbf{w}_t - \alpha\nabla_{\mathbf{w}} h(\mathbf{w}_t, \mathbf{V}_t))$;
13: $\quad\quad \hat{\mathbf{V}}_{t+1} = \mathbf{prox}_{\alpha\lambda\Phi_{\mathbf{V}}}(\mathbf{V}_t - \alpha\nabla_{\mathbf{V}} h(\mathbf{w}_t, \mathbf{V}_t))$;
14: $\quad\quad$ **if** $h(\hat{\mathbf{w}}_{t+1}, \hat{\mathbf{V}}_{t+1}) < h(\bar{\mathbf{w}}_{t+1}, \bar{\mathbf{V}}_{t+1})$ **then**
15: $\quad\quad\quad \mathbf{w}_{t+1} = \hat{\mathbf{w}}_{t+1}$;
16: $\quad\quad\quad \mathbf{V}_{t+1} = \hat{\mathbf{V}}_{t+1}$;
17: $\quad\quad$ **else**
18: $\quad\quad\quad \mathbf{w}_{t+1} = \bar{\mathbf{w}}_{t+1}$;
19: $\quad\quad\quad \mathbf{V}_{t+1} = \bar{\mathbf{V}}_{t+1}$;
20: $\quad\quad$ **end if**
21: $\quad$ **end if**
22: $\quad a_{t+1} = \frac{1}{2}(\sqrt{4a_t^2 + 1} + 1)$
23: $\quad q_{t+1} = \eta q_t + 1$;
24: $\quad c_{t+1} = \frac{1}{q_{t+1}}(\eta q_t c_t + h(\mathbf{w}_{t+1}, \mathbf{V}_{t+1}))$;
25: **end for**
26: **return** $\mathbf{w}_{T+1}, \mathbf{V}_{T+1}$.

4 EXPERIMENTS

In this section, we first introduce the datasets, evaluation metric and experimental settings. And then show the experimental results.

4.1 Datasets

To demonstrate the effectiveness of HIN for recommendation, we conduct experiments on four datasets with rich heterogeneous information. The first dataset is Yelp, which is provided for the Yelp challenge.[1] Yelp is a website where a user can rate local businesses or post photos and reviews about them. The rates fall in the range of 1 to 5, where higher ratings mean users like the businesses while lower rates mean users' negative feedbacks to the businesses. Based on the information collected, the website can recommend businesses according to the users' preferences. Another dataset is Amazon Electronics,[2] which is provided in [8]. As we know, Amazon highly relies on recommendations to present interesting items to users who are surfing on the website. In [8] many domains

[1] https://www.yelp.com/dataset_challenge
[2] http://jmcauley.ucsd.edu/data/amazon/

Table 1: Statistics of Yelp and Amazon datasets.

Yelp-200K				
Relations(A-B)	Number of A	Number of B	Number of (A-B)	Avg Degrees of A/B
User-Business	36,105	22,496	191,506	5.3/8.5
User-Review	36,105	191,506	191,506	5.3/1
User-User	17,065	17,065	140,344	8.2/8.2
Business-Category	22,496	869	67,940	3/78.2
Business-Star	22,496	9	22,496	1/2,499.6
Business-State	22,496	18	22496	1/1,249.8
Business-City	22,496	215	22,496	1/104.6
Review-Business	191,506	22,496	191,506	1/8.5
Review-Aspect	191,506	10	955,041	5/95,504.1
Amazon-200K				
Relations(A-B)	Number of A	Number of B	Number of (A-B)	Avg Degrees of A/B
User-Business	59,297	20,216	183,807	3.1/9.1
User-Review	59,297	183,807	183,807	3.1/1
Business-Category	20,216	682	87,587	4.3/128.4
Business-Brand	95,33	2,015	9,533	1/4.7
Review-Business	183,807	20,216	183,807	1/9.1
Review-Aspect	183,807	10	796,392	4.3/79,639.2

Table 2: The density of rating matrices in the four datasets Density = $\frac{\#\text{Ratings}}{\#\text{Users}\times\#\text{Items}}$.

	Amazon-200K	Yelp-200K	CIKM-Yelp	CIKM-Douban
Density	0.015%	0.024%	0.086%	0.630%

of Amazon dataset are provided, and we choose the electronics domain for our experiments. We extract subsets of entities from Yelp and Amazon to build the HIN, which includes diverse types and relations. The subsets of the two datasets both include around 200,000 ratings in the user-item rating matrices. Thus, we identify them as Yelp-200K and Amazon-200K, respectively. Besides, we also use the datasets provided in the CIKM paper [29], which we call CIKM-Yelp and CIKM-Douban. The statistics of our datasets are shown in Table 1. For the detailed information of CIKM-Yelp and CIKM-Douban, we refer the user to [29]. Note that i) the number of types and relations in the first two datasets, i.e. Amazon-200K and Yelp-200K, we used in this paper are much more than those used in previous works [29, 39, 40]; ii) We give the sparsity of the four datasets in Table 2. The sparsity of the rating matrices is more severe than those used in [29, 39, 40].

4.2 Evaluation Metric

To evaluate the recommendation performance, we adopt the root-mean-square-error (RMSE) as our metric, which is the most popular one for rating prediction in the recommendation literature [14, 20, 22]. RMSE is defined as follows

$$\text{RMSE} = \sqrt{\frac{\sum_{(i,j)\in\mathcal{R}_{test}} (\mathbf{R}_{ij} - \hat{\mathbf{R}}_{ij})^2}{|\mathcal{R}_{test}|}}, \tag{10}$$

where $\mathcal{R}_{test}$ is the set of all user-item pairs (i, j) in the test set, $\hat{\mathbf{R}}_{ij}$ is the predicted rate of user u_i to item b_j, and $\mathbf{R}_{ij}$ is the observed rate of user u_i to item b_j in the test set. For RMSE, smaller value means better performance.

4.3 Baseline Models

We compare the following models to our approach.

- **RegSVD** [15]: RegSVD is the basic matrix factorization with L_2 regularization, which uses only the user-item rating matrix. We run the implementation in Librec [7].[3]
- **FMR** [25]: FMR is the factorization machine with only the user-item rating matrix. We adopt the method in Section 4.1.1 of [25] to model the rating prediction task. We use the code provided by the authors.[4]
- **HeteRec** [40]: HeteRec method is based on meta-path based similarity between users and items. A weighted ensemble model is learned from the latent features of users and items generated by applying matrix factorization to the similarity matrices of different meta-paths. We implemented it based on [40].
- **SemRec** [29]: SemRec is a meta-path based recommendation on weighted HIN, which is built by connecting users and items with the same ratings. Different models are learned from different meta-paths, and a weight ensemble method is used to predict the users' ratings. We use the code provided by the authors.[5]

Note that in [39], the meta-path based similarities are used as regularization terms in the matrix factorization framework. And in [29], the authors reported that SemRec outperforms this method. Thus, we do not report the experimental results of [39] here.

4.4 Experimental Settings

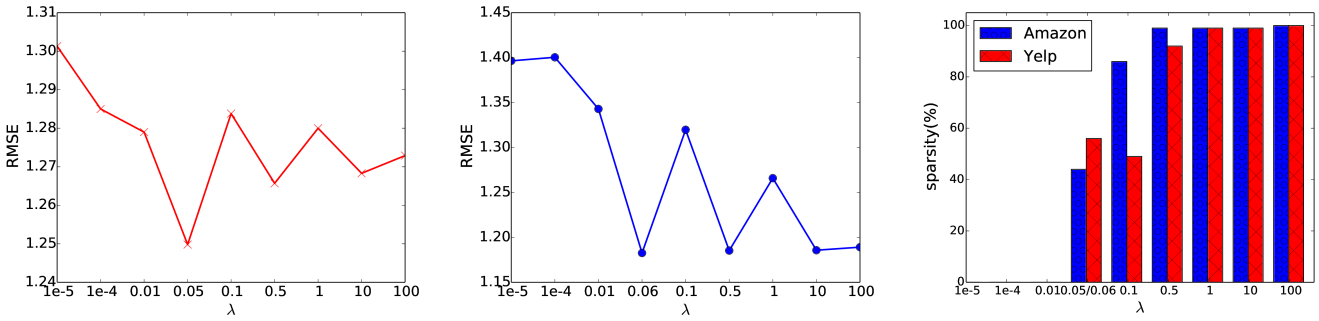
To demonstrate the capability of our model, we use the meta-graphs shown in Figure 3. The meanings of the nodes are given in the figures. To get the aspects from review texts, we use Gensim [24], a topic model software to extract topics. The number of topics is set to 10 empirically. We also try other numbers and they showed similar results. Thus, we fix the number of topics to 10 for all experiments to make fair comparisons.

In terms of designing the experiments, we randomly split the datasets into training and test ones by the ratio 8:2, i.e., 80% of the whole data are used for training and the remaining 20% are for testing. The process is repeated five times and the average RMSE of the five rounds are reported. Our framework is implemented with Python 2.7, and all experiments run in a Linux server with Intel i7 CPU and 32GB RAM.

4.5 Recommendation Effectiveness

We show our results in Table 3. As shown in Table 2, the sparsity of the rating matrices of our data is severe. This is important for the rating prediction task. By comparing RMSEs of Yelp-200K and CIKM-Yelp, we can see that the denser training data results in lower RMSE, i.e., better performance can be obtained. This is because with more observations in the training set, we can get more information about the whole matrix, leading to more accurate predicted ratings of the users to the items. Note that the reason why we did not report the result of SemRec on Amazon-200K is that the programs

[3] https://www.librec.net/
[4] http://www.libfm.org/
[5] https://github.com/zzqsmall/SemRec



(a) Yelp. (b) Amazon. (c) Sparsity of w, V.

Figure 4: Effects of λ. (a) and (b) show RMSEs with different λ's. (c) shows the trend of sparsity. Note that at the fourth point of x-axis we have $\lambda = 0.05$ for Yelp and $\lambda = 0.06$ for Amazon, respectively.

Table 3: Recommending performance in terms of RMSE. Percentages in the brackets are the reduction of RMSE comparing our approach with the corresponding approaches in the table header.

	Amazon-200K	Yelp-200K	CIKM-Yelp	CIKM-Douban
RegSVD	2.9656 (+60.0%)	2.5141 (+50.5%)	1.5323 (+27.7%)	0.7673 (+9.0%)
FMR	1.3462 (+11.9%)	1.7637 (+29.4%)	1.4342 (+22.8%)	0.7524 (+7.2%)
HeteRec	2.5368 (+53.2%)	2.3475 (+47.0%)	1.4891 (+25.6%)	0.7671 (+9.0%)
SemRec	- -	1.4603 (+14.7%)	1.1559 (+4.2%)	0.7216 (+3.2%)
FMG	**1.1864**	**1.2456**	**1.1074**	**0.6985**

crashed on Amazon-200K in a server with 128G memory due to the large numbers of users and items as shown in Table 1.

From Table 3, we can see that comparing to RegSVD and FMR, which only use the rating matrix, SemRec and FMG, which use additional heterogeneous information by meta-graphs, are significantly better. Especially, the sparser the rating matrix, the more useful the additional information incorporated. For example, on Amazon-200K, FMG outperforms RegSVD by 60%, while for CIKM-Douban, the percentage of RMSE decreasing is 9%. Note that the performance of HeteRec is worse than FMR, despite the fact that we have tried our best to tune the models. The reason is that, as we show in Section 2.3, using a weighting ensemble of dot product of latent features may lose information among the meta-graphs and fail to avoid noise caused by too many meta-graphs.

When comparing the results of FMG and SemRec, we find that the performance gap between them are not that large, which means that SemRec is still a good method for rating prediction, especially when comparing SemRec to the other three baselines. The good performance of SemRec may be attributed to two reasons. First, incorporating rating values into HIN leads to a weighted HIN, which may better capture the meta-graph or meta-path based similarity. Currently, FMG ignores the rating values, so it remains unknown whether it can further decrease RMSE if we adopt a similar approach to incorporate rating values into HIN. We leave it as future work. Second, the meta-graphs SemRec exploits are all of the style like $U \rightarrow * \leftarrow U \rightarrow B$, which have a good capability of predication. In the Section 4.6, we will show that FMG can automatically select features constructed by meta-graphs like $U \rightarrow * \leftarrow U \rightarrow B$ while removing those by meta-graphs like $(U \rightarrow B \rightarrow * \leftarrow B)$. In Section 4.7, we further study the prediction ability of each meta-graph, and also show that meta-graphs with style like $U \rightarrow * \leftarrow U \rightarrow B$ are better than those like $U \rightarrow B \rightarrow * \leftarrow B$.

4.6 The Parameter λ

In this part, we show the influence of parameter λ, with $\lambda = \lambda_w = \lambda_v$, which controls the effects of group lasso. The experiments were conducted on Yelp-50k and Amazon-50k, where only 50,000 ratings are sampled and thus is a smaller version of Yelp-200K and Amazon-200K for the sake of efficiency of parameter tuning. The RMSEs of Yelp-50k and Amazon-50K are shown in Figures 4(a) and (b), respectively. We can see that with λ increasing, RMSE decreases first and then increases, demonstrating that λ values that are too large or too small are not good for the performance of rating prediction. Specifically, on Yelp, the best is $\lambda = 0.05$, and on Amazon, the best is $\lambda = 0.06$. Next, we give further analysis of these two parameters in terms of sparsity and the selected meta-graphs by group lasso.

Sparsity of $\mathbf{w}, \mathbf{V}$. We now study the sparsity of the learned parameters, i.e., ratio of zeros in $\mathbf{w}, \mathbf{V}$ after learning. Sparsity is defined as sparsity $= \frac{z}{w_n + v_n}$, where z is the total number of zeros in $\mathbf{w}$ and $\mathbf{V}$, and w_n and v_n are the number of entries in $\mathbf{w}$ and $\mathbf{V}$, respectively. The larger sparsity is, the more zeros are in $\mathbf{w}$ and $\mathbf{V}$, which will reduce the time of online prediction. The trend of sparsity with different λ's is shown in Figure 4(c). We can see that with λ increasing, the sparsity becomes greater, which aligns with the effects of group lasso. Note that the trend is non-monotonous due to the non-convexity of the objective function w.r.t. $\mathbf{w}$ and $\mathbf{V}$ and the fact that we set $\lambda_w = \lambda_v$ for the convenience of parameter tuning.

The Selected Meta-graphs. In this part, we analyze the selected features by group lasso. From Figure 4, we can see that $\mathbf{w}$ and $\mathbf{V}$

Table 4: Selected Meta-graphs for Yelp and Amazon datasets.

		User-Part		Item-Part	
		w	**V**	**w**	**V**
Yelp	Important	$M_1 - M_4, M_6, M_8$	$M_1 - M_3, M_5, M_8$	$M_1 - M_5, M_8, M_9$	M_3, M_8
	Useless	M_5, M_7, M_9	M_4, M_6, M_7, M_9	M_6, M_7	$M_1, M_2, M_4 - M_7, M_9$
Amazon	Important	$M_1 - M_3, M_5$	$M_1 - M_6$	M_2, M_3, M_5, M_6	M_2, M_5, M_6
	Useless	M_4, M_6	-	M_1, M_4	M_1, M_3, M_4

are good in terms of RMSE and sparsity when $\lambda = 0.05$ on Yelp and $\lambda = 0.06$ on Amazon. Thus, we show the most important meta-graphs with this configuration for user and item latent features. The results of Amazon and Yelp are shown in Table 4.

From Table 4, we can observe that both the first-order and second-order interactions are important for overall performance, which demonstrates that the second-order interactions are necessary for better recommendation performance. As we mentioned in Section 1, previous works do not fully make use of the latent features, like the second-order interactions.

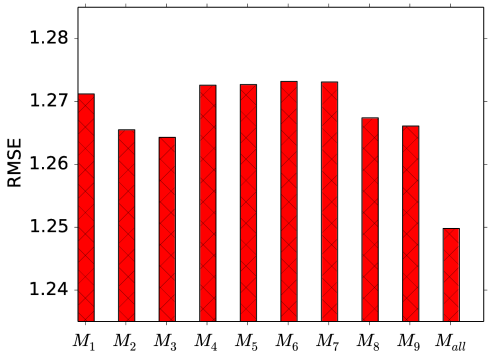
Another discovery is that the meta-graphs with style like $U \rightarrow * \leftarrow U \rightarrow B$ are better than those like $U \rightarrow B \rightarrow * \leftarrow B$. Here we use $U \rightarrow * \leftarrow U \rightarrow B$ to represent meta-graphs like M_2, M_3, M_8, M_9 in Figure 3(a) and M_2, M_5, M_6 in Figure 3(b), while use $U \rightarrow B \rightarrow * \leftarrow B$ to represent meta-graphs like M_4, M_5, M_6, M_7 in Figure 3(a) and M_3, M_4 in Figure 3(b). For Yelp, we can see that meta-graphs like M_2, M_3, M_8, M_9 tend to be selected while $M_4 - M_7$ are removed, which means that on Yelp, recommendations by friends or similar users are better than those by similar items. Similar cases exist on Amazon, i.e., M_3, M_4 tend to be removed.

Finally, on both datasets, complex structures like M_9 in Figure 3(a) and M_6 in Figure 3(b) are determined to be important for item latent features, which demonstrates the importance of capturing these kinds of relations, which are ignored by previous meta-path based recommendation methods [29, 39, 40].

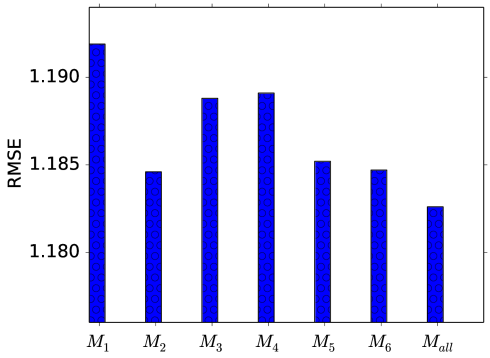
4.7 Recommending Performance with Single Meta-Graph

In this part, we compare the performances of different meta-graphs separately. In the training process, we use only one meta-graph for user and item features and then predict and evalute the results obtained by the corresponding meta-graph. Specifically, we run experiments to compare RMSEs of all of the meta-graphs in Figure 3. The RMSE of each meta-graph is shown in Figure 5. Note that we show as comparison the RMSEs of all the meta-graphs used, denoted by M_{all}.

From Figure 5, we can see that on both Yelp and Amazon, the performances are the best when all meta-graph based user and item features are used, which demonstrates the usefulness of the semantics captured by the designed meta-graphs in Figure 3. Besides, we can see that on Yelp, the performances of $M_4 - M_7$ are the worst, and on Amazon, the performances of $M_3 - M_4$ are also among the worst three. Note that they are both meta-graphs with style like $U \rightarrow B \rightarrow * \leftarrow B$. Thus, it aligns with the observation in Section 4.6 that meta-graphs with style like $U \rightarrow * \leftarrow U \rightarrow B$ are better than those like $U \rightarrow B \rightarrow * \leftarrow B$.



(a) Yelp.



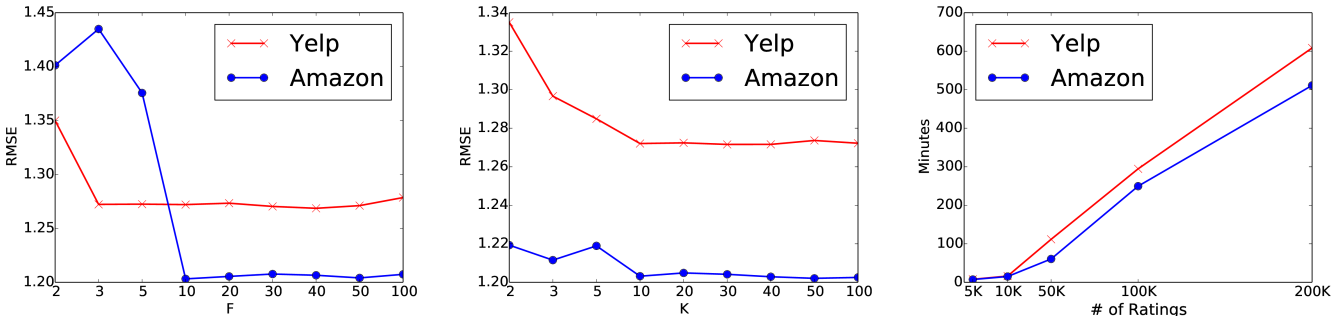
(b) Amazon.

Figure 5: RMSE of single meta-graph on Yelp and Amazon datasets. M_{all} is our model trained with all meta-graphs.

Finally, for M_9 on Yelp and M_6 on Amazon, we can see that the performances of these two meta-graphs are among the best three, which demonstrates the usefulness of the complex semantics captured in M_9 on Yelp and M_6 on Amazon. In future work, we will try to design more complex meta-graphs like these two, and study if they can further improve recommending performance.

4.8 The Parameters F and K

In this part, we study the influence of the two parameters: F and K. F is the rank used to factorize meta-graph based user-item similarity matrices to obtain user and item latent features (see



(a) Varying F with $K = 10$. (b) Varying K with $F = 10$. (c) Scalability.

Figure 6: (a) and (b) show RMSEs with different K's and F's, (c) shows the execution time with different sizes of datasets.

Section 2.2). K is the number of factors to factorize the second-order weights $\mathbf{V}$ in the FMG model (see Section 2.3). For the sake of efficiency, we conduct extensive experiments using the smaller datasets, Yelp-50K and Amazon-50K. We set F and K to values in the range of $[2, 3, 5, 10, 20, 30, 40, 50, 100]$. The results are shown in Figures 6(a) and (b). We can see that the performances get better with larger F or K in both datasets. After a threshold value, i.e., 10, the performance becomes stable. In practice, the performance gains are marginal when the two parameters are greater than 10, and larger values mean higher computational cost. Therefore, it is enough to obtain a good performance to set $F = 10$ and $K = 10$ for these two datasets, which are the adopted settings of these two parameters of the experimental results reported in Section 4.5.

4.9 Scalability

In this part, we study the scalability of our FMG model. We extract a series of datasets of different scales from Yelp-200K and Amazon-200K according to the number of observations in the user-item rating matrix. The specific values are $[5K, 10K, 50K, 100K, 200K]$. The time cost of Amazon and Yelp datasets are shown in Figure 6(c), for which we set $\lambda = 0.05$ for Yelp and $\lambda = 0.06$ for Amazon and number of iterations to $3,000$. We can see from the figure that the training time is almost linear to the number observed ratings.

5 RELATED WORK

In this section, we briefly introduce the related work of HINs and recommendation.

5.1 Heterogeneous Information Networks

HINs have been proposed as a general representation for many real-world graphs or networks. Meta-path has been developed as a sequence of entity types defined by the HIN network schema. Based on a meta-path, several similarity measures, such as PathCount [32], PathSim [32], and Path Constrained Random Walk [16] have been proposed. These measures have been shown to be useful for entity search and similarity measure in many real networks. After the development of meta-path, many data mining tasks have been enabled or enhanced including recommendation [29, 39, 40], similarity search [27, 28, 32], clustering [33, 35], classification [1, 12, 18], and link prediction [31, 43]. Recently, meta-graph (or meta-structure) has been proposed to define more complicated semantics in HIN [6, 10]. They still applied meta-graph to entity similarity problem where entities are constrained to be of the same type. In this paper, we extend this idea to recommendation problem. The problem of recommendation requires us to approximate the large-scale user-item rating matrix. Thus, instead of computing each similarity efficiently online, we consider to compute the matrices offline, and design the best way to use the user-item matrices generated by different meta-graphs for the final prediction.

5.2 Recommendation in HIN

Modern e-commerce websites allow us to incorporate heterogeneous information in making recommendations. Traditional recommendation systrems must be enhanced to make use of the rich semantics provided by the heterogeneous information. For example, Ma et al. [20] incorporated social relations as regularization sterm into the matrix factorization in recommendation systems. In [4, 34], items' meta-data are modeled to improve the recommendation task. In [19, 21], the review texts are analyzed together with the ratings in the rating prediction task. Ye et al. [38] proposed a probabilistic model to incorporate users' preferences, social network and geographical information to enhance the point-of-interests recommendation. These previous approaches have demonstrated the importance and effectiveness of heterogeneous information in improving recommendation accuracy. However, most of these approaches dealt with different heterogeneous information disparately, hence losing important information that exist across them.

HIN-based recommendation has been proposed to avoid the disparate treatment of different types of information. Based on meta-path, several approaches have attempted to tackle the recommendation task based on HIN. In [39], meta-path based similarities are used as regularization terms in the matrix factorization framework. In [40], multiple meta-paths are used to learn user and item latent features, which are then used to recover similarity matrices combined by a weighted mechanism. In [29], users' ratings to items are used to build a weighted HIN, based on which meta-path based methods are used to measure the similarities of users for recommendation. The combination of different meta-paths are explicit, using the similarities instead of latent features. As discussed in the introduction, all of the above approaches do not

make full use of the meta-path based features, whereas our approach based on the factorization machine can do.

5.3 Factorization Machine

Factorization Machine [25] is a state-of-the-art recommendation framework, which can model the interactions among features, e.g., the rating information, categories of items, texts, time, etc. Therefore, it is a powerful framework to integrate content features for collaborative-filtering-based recommendation. Many approaches and systems have been developed based on FMs [9, 11, 26]. Different from previous approaches which only consider explicit features, we generate latent features by matrix factorization based on different meta-graphs. For FM using the original explicit features, MF can be regarded as a step similar to PCA to perform dimensionality reduction to reduce the noise of the original features.

6 CONCLUSION

In this paper, we present a heterogeneous information network (HIN) based recommendation method. We introduce a principled way of fusing heterogeneous information in the network. By using meta-graphs derived from the HIN schema, we can formulate complicated semantics between users and items. Then, we use matrix factorization to obtain latent features of user and item from each meta-path in an unsupervised way. After that, we use a group lasso regularized factorization machine to fuse different groups of semantic information extracted from different meta-graphs to predict the links. Experimental results demonstrate the effectiveness of our approach. In the future, we plan to explore richer information to enrich the features and semantics in the network, and use parallel computing and deep learning to further improve our system.

7 ACKNOWLEDGMENT

Research in this paper was partially supported by the Research Grants Council HKSAR GRF (No.615113 and No.614513), China 973 Fundamental R&D Program (No.2014CB340304) and Hong Kong ITF Grant (No. ITF/391/15FX). We also thank the anonymous reviewers for their valuable comments and suggestions that help improve the quality of this manuscript.

REFERENCES

[1] P. Bangcharoensap, T. Murata, H. Kobayashi, and N. Shimizu. Transductive classification on heterogeneous information networks with edge betweenness-based normalization. In *WSDM*, pages 437–446, 2016.
[2] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.
[3] J. C. D. Chapter 13: Mining electronic health records in the genomics era. *PLoS Computational Biology*, 8(12), 2012.
[4] A. Deepak and B. Chen. fLDA: Matrix factorization through latent dirichlet allocation. In *WSDM*, pages 91–100, 2010.
[5] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*, pages 601–610, 2014.
[6] Y. Fang, W. Lin, V. W. Zheng, M. Wu, K. C.-C. Chang, and X. Li. Semantic proximity search on graphs with meta graph-based learning. In *ICDE*, pages 277–288, 2016.
[7] G. Guo, J. Zhang, Z. Sun, and N. Yorke-Smith. Librec: A Java library for recommender systems., 2015.
[8] R. He and J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, pages 507–517, 2016.
[9] L. Hong, A. S. Doumith, and B. D. Davison. Co-factorization machines: Modeling user interests and predicting individual decisions in twitter. In *WSDM*, pages 557–566, 2013.
[10] Z. Huang, Y. Zheng, R. Cheng, Y. Sun, N. Mamoulis, and X. Li. Meta Structure: Computing relevance in large heterogeneous information networks. In *KDD*, pages 1595–1604, 2016.
[11] Y. Juan, Y. Zhuang, W. Chin, and C. J. Lin. Field-aware factorization machines for CTR prediction. In *RecSys*, pages 43–50, 2016.
[12] X. Kong, B. Cao, and P. S. Yu. Multi-label classification by mining label and instance correlations from heterogeneous information networks. In *KDD*, pages 614–622, 2013.
[13] X. Kong, J. Zhang, and P. S. Yu. Inferring anchor links across multiple heterogeneous social networks. In *CIKM*, pages 179–188, 2013.
[14] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434, 2008.
[15] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems, 2009.
[16] N. Lao and W. W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81(1):53–67, 2010.
[17] H. Li and Z. Lin. Accelerated proximal gradient methods for nonconvex programming. In *NIPS*, pages 379–387, 2015.
[18] X. Li, B. Kao, Y. Zheng, and Z. Huang. On transductive classification in heterogeneous information networks. In *CIKM*, pages 811–820, 2016.
[19] G. Ling, M. R. Lyu, and I. King. Ratings meet reviews, a combined approach to recommend. In *RecSys*, pages 105–112, 2014.
[20] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *WSDM*, pages 287–296, 2011.
[21] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, pages 165–172, 2013.
[22] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2007.
[23] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.
[24] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora, May 2010.
[25] S. Rendle. Factorization machines with libFM. *ACM Transactions on Intelligent Systems and Technology*, 3(3):57:1–57:22, 2012.
[26] S. Rendle and L. S. T. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*, pages 81–90, 2010.
[27] C. Shi, X. Kong, Y. Huang, Y. P. S., and B. Wu. HeteSim: A general framework for relevance measure in heterogeneous networks. *IEEE Transactions on Knowledge and Data Engineering*, 26(10):2479–2492, 2014.
[28] C. Shi, X. Kong, P. S. Yu, S. Xie, and B. Wu. Relevance search in heterogeneous networks. In *ICEDT*, pages 180–191, 2012.
[29] C. Shi, Z. Zhang, P. Luo, P. S. Yu, Y. Yue, and B. Wu. Semantic path based personalized recommendation on weighted heterogeneous information networks. In *CIKM*, pages 453–462, 2015.
[30] Y. Sun and J. Han. Mining heterogeneous information networks: a structural analysis approach. *ACM SIGKDD Explorations Newsletter*, 14(2):20–28, 2013.
[31] Y. Sun, J. Han, C. C. Aggarwal, and N. V. Chawla. When will it happen?: Relationship prediction in heterogeneous information networks. In *WSDM*, pages 663–672, 2012.
[32] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. PathSim: Meta path-based top-k similarity search in heterogeneous information networks. *VLDB*, 4(11):992–1003, 2011.
[33] Y. Sun, B. Norick, J. Han, X. Yan, P. S. Yu, and X. Yu. PathSelClus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. *ACM Transactions on Knowledge Discovery from Data*, 7(3):11, 2013.
[34] F. Vasile, E. Smirnova, and A. Conneau. Meta-Prod2Vec: Product embeddings using side-information for recommendation. In *RecSys*, pages 225–232, 2016.
[35] C. Wang, Y. Song, A. El-Kishky, D. Roth, M. Zhang, and J. Han. Incorporating world knowledge to document clustering via heterogeneous information networks. In *KDD*, pages 1215–1224, 2015.
[36] Q. Yao and J. Kwok. Accelerated inexact soft-impute for fast large-scale matrix completion. In *AAAI*, pages 4002–4008, 2015.
[37] Q. Yao and J. Kwok. Efficient learning with a family of nonconvex regularizers by redistributing nonconvexity. In *ICML*, pages 2645–2654, 2016.
[38] M. Ye, P. Yin, W. C. Lee, and D. L. Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *SIGIR*, pages 325–334, 2011.
[39] X. Yu, X. Ren, Q. Gu, Y. Sun, and J. Han. Collaborative filtering with entity similarity regularization in heterogeneous information networks. *IJCAI HINA*, 27, 2013.
[40] 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*, pages 283–292, 2014.
[41] L. Yuan, J. Liu, and J. Ye. Efficient methods for overlapping group lasso. In *NIPS*, pages 352–360, 2011.
[42] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B*, 68(1):49–67, 2006.
[43] J. Zhang, P. S. Yu, and Z. Zhou. Meta-path based multi-network collective link prediction. In *KDD*, pages 1286–1295, 2014.