Unsupervised P2P Rental Recommendations via Integer Programming

Yanjie Fu
Missouri Univ. of Sci. and Tech.
Missouri, USA
fuyan@mst.edu

Guannan Liu*
Beihang University
Beijing, China
liugn@buaa.edu.cn

Mingfei Teng
Rutgers University
NJ, USA
tengmf@gmail.com

Charu Aggarwal
IBM T. J. Watson Research Center
NY, USA
charu@us.ibm.com

ABSTRACT

Due to the sparseness of quality rating data, unsupervised recommender systems are used in many applications in Peer to Peer (P2P) rental marketplaces such as Airbnb, FlipKey, and HomeAway. We present an integer programming based recommender systems, where both accommodation benefits and community risks of lodging places are measured and incorporated into an objective function as utility measurements. More specifically, we first present an unsupervised fused scoring method for quantifying the accommodation benefits and community risks of a lodging with crowd-sourced geotagged data. In order to the utility of recommendations, we formulate the unsupervised P2P rental recommendations as a constrained integer programming problem, where the accommodation benefits of recommendations are maximized and the community risks of recommendations are minimized, while maintaining constraints on personalization. Furthermore, we provide an efficient solution for the optimization problem by developing a learning-to-integer-programming method for combining aggregated listwise learning to rank into branching variable selection. We apply the proposed approach to the Airbnb data of New York City and provide lodging recommendations to travelers. In our empirical experiments, we demonstrate both the efficiency and effectiveness of our method in terms of striving a trade-off between the user satisfaction, time on market, and the number of reviews, and achieving a balance between positive and negative sides.

KEYWORDS

Unsupervised Recommendations; Integer Programming; Learning To Optimize;

*Contact author.

KDD'17, August 13–17, 2017, Halifax, NS, Canada.
 ISBN 978-1-4503-4887-4/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3097983.3098044

1 INTRODUCTION

The sharing economy has provided travelers more choices beyond traditional hotels with the emergence of Internet-enabled P2P rental markets, e.g., Airbnb (www.airbnb.com), FlipKey (www.flipkey.com), HomeAway (www.homeaway.com), where local residents serve as hosts to provide home stay while travelers search, identify, and compensate their hosts. Although current P2P platforms bring cheaper lodging options for travelers, it is however a very challenging task for the travelers to find satisfactory lodging places in an unfamiliar city due to the following issues: (1) Lodgings are on or off markets dynamically, and thus are lack of quality rating data. For example, some apartments are listed or rented only once. (2) Lodgings are mostly private properties without location planning and hospitality management for travel accommodation purposes. For example, some places may not be convenient for entertainment and sightseeing, or the amenities may not be well equipped. (3) Travelers have little knowledge about the neighborhood environments of the lodgings listed on the markets. For example, some places are located in unsafe neighborhoods with many sexual assaults, gunshots, and thefts.

To facilitate travelers to search lodgings, today's P2P rental platforms provide rankings of posted lodgings, typically using filtering criteria such as price, amenities, review ratings, etc. However, existing approaches are far from adequate due to the following reasons. Since travelers are unfamiliar with destination cities, they may have to probe into as many listings as possible by browsing related review comments from third-party sources, checking the neighborhoods of interested places with Google Map, and communicating with hosts. These may cost tremendous time and efforts, and thus it highly necessitates a smart system to recommend appropriate lodging places, to overcome the lack of quality user ratings, and moreover, to alleviate the information asymmetry between travelers and hosts in the P2P platforms.

There are several challenges in achieving the goal inherent in our system. First, it is difficult to estimate the accommodation benefits and community risks of P2P rentals as they are privately owned and dynamically listed, with limited information publicly available. With the advent of mobile, sensing and Internet technologies, the availability of mobile data including urban geography (e.g., POIs), human mobility (e.g., GPS traces), and public service data (e.g., crimes, noises) are ever increasing. These data can be a

source of rich intelligence for scoring accommodation benefits and community risk of lodgings. However, how to score accommodation benefits and community risks of lodgings under unsupervised settings by fusing heterogeneous data is a challenging problem.

Second, a P2P rental recommender system can be viewed in terms of benefit and risk. In other words, the system should have the capability of maximizing accommodation benefits and minimizing community risks. Moreover, users also have personalized constraints. For instance, the cost of a recommended place should be within the user's budget; the availability of a recommended place should be long enough to cover the planed travel days. Therefore, how can we simultaneously achieve multiple objectives? Unlike traditional supervised recommendation algorithms such as collaborative filtering and matrix factorizations, we alternatively formulate this problem as a constrained *I*nteger *P*rogramming (IP), where the utility of recommendations is maximized while constraints are for personalization, in order to navigate these trade-offs in a flexible and meaningful way.

Third, a natural approach for the IP based recommendation problem is the *B*ranch and *B*ound (BB) method. However, traditional BB methods exhaustedly examine every candidate branching variable, in order to find the best branching variable to reduce the gap between the best bound and the current best feasible solution value, and thus are computationally expensive. We adapt the idea of learning by analogy. In reality, it is often possible to collect many other data sets, which have labels or ratings provided by domain experts. Running BB on these labeled datasets can enable us to observe how BB selects branching variables and computes corresponding gains of different branching events. The collected information (i.e., branching variable selection and gains) of BB on labeled datasets can provide a "meta training" collection of the branching behaviors of BB, which can be used to identify the key statistical characteristics of the best branching variables that help BB find the optimal solutions faster.

Along these lines, in this paper, we develop an unsupervised framework for recommending lodging places with awareness of benefit and safety. Specifically, we first extract the features of accommodation benefits for each lodging from urban geography, human mobility, and property amenities data. Besides, we mine the features of community risk for each lodging from the publicly available data of crimes, noises, and complains. In addition, we devise an unsupervised fused scoring method to select, weight, and aggregate the corresponding features into benefit and risk scores for each lodging. Moreover, we construct an optimization objective for recommending lodgings by simultaneously maximizing accommodation benefit and minimizing community risk with respect to personalized constraints. We also develop a learning to IP method by presenting an aggregated listwise branching selector. Finally, we present extensive experiments to demonstrate the enhanced performance of the proposed method using real world P2P rental market data.

2 PRELIMINARIES

In this section, we first introduce the unsupervised recommendations formulation for P2P rental markets, and then detail how to measure accommodation benefits and community risks of a lodging for constructing the utility of a lodging.

2.1 Motivation and Problem Statement

The decision process of finding a lodging place in a P2P rental market is different from that of booking a starred hotel. Travelers not only need to consider personalized constraints (i.e., cost budget, length of stay), but also take into account both the positive and negative factors (i.e., accommodation benefits and community risks), which have been considered in starred hotels but not in P2P rental markets. Formally, considering a set of candidate lodgings, $I = \{1, 2, 3, ..., |I|\}$, let $i \in I$ be the index of a candidate lodging, with $u_i \in U$ being its accommodation benefit, $r_i \in R$ being its community risk, we aim to determine K lodgings as a recommendation set, indicated by a binary decision vector $\mathbf{y}$, by maximizing accommodation benefit, minimizing community risk, while subject to personalized constraints.

We propose to formulate this task as an integer programming as shown in Definition 2.1, where (i) the criteria in terms of benefit and risk is denoted as Discounted Accumulated Gain, DCG($\mathbf{y}$, U, R), which is in a form of weighted sum of benefits and risks of recommended lodgings. (ii) the criteria in terms of personalized constraints is directly formulated as constraints in the integer programming problem.

Definition 2.1. **Problem Statement.** *Given a set of candidate lodgings, I, each with a benefit measurement and a risk measurement, recommend K lodgings such that the discounted accumulated gain, i.e., DCG($\mathbf{y}$, U, R), is maximized, while respecting personalized constraints.*

Specifically, each lodging has a normalized benefit score and a normalized risk score, i.e., $DCG(\mathbf{y}, U, R) = \sum_{i \in I} y_i(\lambda u_i - r_i)$. Also, the personalized constraints can be instantiated by two commonly used constraints in lodging search, i.e., cost budget and length of stay. However, in addition to cost and length of stay, the model can be easily extended to more personalized constraints for more sophisticated personalized recommendations. Accordingly, the lodging recommendation problem can be further formulated as follows:

$$\max_{\mathbf{y}} \mathcal{F}(\mathbf{y}) = \sum_{i \in I} y_i(\lambda u_i - r_i) \quad (1)$$

$$s.t. \quad y_i \in \{0, 1\} \quad (2)$$

$$\forall p \in \{i | y_i = 1\} : c_p \leq C \quad (3)$$

$$\forall q \in \{i | y_i = 1\} : t_q \geq T^1 \quad (4)$$

$$\sum_{i \in I} y_i = K \quad (5)$$

where λ is a parameter representing the weight of accommodation benefits compared to community risks. In Constraint (2), $\mathbf{y} = \{0, 1\}^{|I|}$ is a binary vector. $y_i = 1$ indicates that the lodging place i is selected as a recommended lodging place, otherwise $y_i = 0$. Constraint (3) simply states that the maximum cost budget is C. Constraint (4) simply states that the minimum length of stay is T. Constraint (5) states that the number of recommended lodging places is K.

[1]In reality, a user usually input start date and end date for lodging search and reservation. To simplify the formulation of objective function and data processing, we use single input: length of stay. This simplified objective function can be easily adapted for two inputs: start date and end date.

Table 1: Benefit-related features.

Categories	Source	Variable Name	Variable Description
Public Transportation	NYC Taxi Commission	reachability	number of bus stops one can reach by bus in one hour
		transit density	number of bus stops or subway stations in the neighboring circle area of radius one kilometer
		distance to transit	walking distance to the nearest bus stop or subway station
POIs	Foursquare NYC POIs	POI frequency	number of POIs in terms of different POI categories (e.g., shopping, foods, education, entertainment, etc.)
		POI diversity	entropy given POI frequency density over different POI categories
Mobility	Foursquare NYC Checkins	neighborhood popularity	number of smart phone checkins when people visit outdoor places
Space and Amenities	Airbnb NYC	bath rooms	numbers of bath rooms
		bedrooms	numbers of bedrooms
		beds	numbers of beds
		kitchen	binary indicator of kitchen
		Internet	binary indicator of Internet
		TV	binary indicator of TV
		air conditioner	binary indicator of air conditioner
		heat	binary indicator of heat

Table 2: Risk-related features.

Categories	Source	Variable Name	Variable Description
Crime	NYPD	crime frequency	number of criminal events over different offense categories (burglary, felony assault, grand larceny of motor vehicle, murder, rape, robbery)
		crime diversity	entropy given criminal frequency density over different offense categories
Complaints	NYC 311 Service	complaint frequency	complaint frequencies over different complaint types (air, animal, decoration, dirty, electricity, facility, food, human, noise, safety, transport, water)
		complaint diversity	entropy given complaint frequency density over different complaint types

2.2 Unsupervised Scoring of Accommodation Benefits and Community Risks

We first construct benefit-related and risk-related features from heterogeneous lodging related information, and then develop an unsupervised fused scoring method to score the accommodation benefit and community risk of each lodging.

Construction of Benefit-related and Risk-related Features. First, accommodation benefits can be assessed from positive sides based on urban geography, human mobility, and property amenities. Specifically, urban geography data such as public transportations and POIs describe the neighborhood infrastructure near a place; human mobility data such as vehicle GPS traces and check-ins encode the geographic preferences of mobile users and the popularity of a place. Moreover, the information such as space and amenities of a listed lodging is usually specified on the website, which represents the comfortableness and functionality and can be incorporated to estimate the benefit. Table 1 lists the features to depict of accommodation benefits. Second, community risk can be perceived from negative sides of the neighborhoods such as crimes, complains, and noises, which have been publicly available from local government websites. Table 2 lists the features of community risks.

Unsupervised Fused Scoring. We propose a three-step unsupervised method to aggregate the features of benefits (risks) into a single benefit (risk) score. The challenge mainly lies on how to effectively select discriminative features and weigh feature distinctiveness under an unsupervised setting.

Step1: Unsupervised Feature Selection. Features can be selected to be mutually far away from each other such that the information redundancy is minimized. To measure the feature redundancy, we use mutual information for discrete features. Given two features X and Y, the mutual information is computed by $\rho(X,Y) = \sum_{y\in Y}\sum_{x\in X} p(x,y)log(\frac{p(x,y)}{p(x)p(y)})$, where $p(x,y)$ is the joint distribution of X and Y, and $p(x)$ and $p(y)$ are marginal distributions of X and Y, respectively. For continuous features, we use statistical correlation (i.e., Pearson product-moment correlation coefficient) which is given by $\rho(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$, where cov is the covariance and σ_X is the standard deviation of X. Then, we propose a fast algorithm to select non-redundant features. The idea is as follows: we first compute the pair-wise correlation between features, and identify highly-correlated feature pairs, i.e., pairs with absolute values of correlation larger than a threshold α (set to 0.9). From the two features in each identified feature pair, we remove the feature with larger mean absolute correlation (computed as the average of the absolute value of the Pearson correlation of this feature with all other features). By this selection procedure, we are able to obtain a set of features with low redundancy.

Step2: Unsupervised Feature Weighting. The weight of a feature represents the discrimination capability of classifying accommodation benefits or community risks. We propose a Laplacian Score [10] based approach to quantify the weight of each feature. The Laplacian score for a feature is computed based on Laplacian Eigenmaps and Locality Preserving Projection, in order to capture its classification power. Consider a data table $\mathbf{F} \in \mathbb{R}^{P\times Q}$ with P instances and Q features, where the instance $\mathbf{f}_{p,*}$ is the p-th row in $\mathbf{F}$ and the feature $\mathbf{f}_{*,q}$ is the q-th column. To extract the Laplacian Score of the feature q, we first construct an undirected graph with nodes as instances (i.e., rows). Each edge $(p1, p2)$ is associated with a weight representing the Cosine similarity between the row $\mathbf{f}_{p1,*}$ and the row $\mathbf{f}_{p2,*}$. Thereby, we can use a weight matrix $\mathbf{S} \in \mathbb{R}^{P\times P}$ to describe this graph and represent the local structure of the data space F. Finally, the Laplacian Score of the q-th feature is defined by $L_q = \frac{\sum_{p1=1}^{P}\sum_{p2=1}^{P}(f_{p1,q}-f_{p2,q})^2 s_{p1,p2}}{Var(\mathbf{f}_{*,q})}$, where $f_{p1,q}$ is the value at $p1$-th row and q-th column in $\mathbf{F}$, $Var(\mathbf{f}_{*,q})$ is the variance of the q-th column, and $s_{p1,p2} \in \mathbf{S}$.

Step3: Normalization, Aggregation and Scoring. Given features and corresponding weights, we normalize features using a quotient transformation (x/max). Also, we normalize feature weights using another quotient transformation (x/sum). Finally, we combine features and weights into a single score in a form of weighted sum of components. We will apply this method to the benefit feature set and the risk feature set, and extract two corresponding scores for each lodging.

3 LISTWISE LEARNING TO INTEGER PROGRAMMING

We present a learning to integer programming method as an effective solution for the optimization problem, by combining listwise learning to rank and learning by analogy into branch and bound.

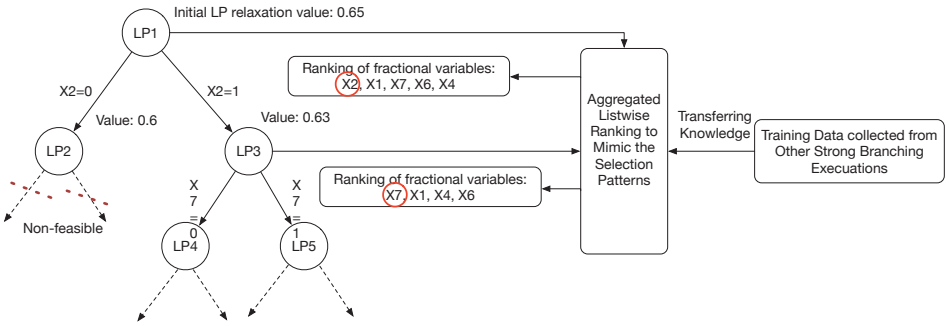


Figure 1: Incorporating Aggregated Listwise Branching Variable Selector into BB to Transfer Knowledge from Strong Branching

3.1 Background of Branch-and-Bound

Compared to probabilistic searching (e.g., genetic algorithm or simulated annealing), BB can solve this objective in a fast, accurate and stable manner. BB begins by finding the optimal solution to the relaxation of the problem without the integer constraints via standard linear programming. If in this solution, the decision variables with integer constraints have integer values, then no further work is required. If one or more integer variables have non-integral solutions, the method chooses one such variable and "branches", creating two new subproblems where the value of that variable is more tightly constrained. These subproblems are solved and the process is repeated, until a solution that satisfies all of the integer constraints is found.

Therefore, it is critical to select proper branching variables effectively and efficiently in BB. One widely-used strategy is strong branching [1]: if a solution E is not integer feasible, let V be the set of candidate branching variables in E with objective value z. For all candidate branching variables $j \in V$, we compute a branching gain g_j as follows: assuming branching E on j downwards and upwards, the two solutions E^+ and E^- of two newly created subproblems have feasible objective values z_j^+ and z_j^-, then the gain g_j for the branching variable j is defined as $g_j = (z_j^+ - z) \times (z_j^- - z)$. Strong branching strategy will identify the best branching variable with maximum gain: $\max_{j \in V} g_j$.

3.2 An Aggregated Listwise Branching Selector

Traditional BB, including strong branching, is a time-consuming process due to its exhaustive strategy. To strive a balance between efficiency and effectiveness, we improve BB by replacing strong branching with a new branching variable selector. We adapt the idea of learning by analogy. It is often possible to collect many other data sets with labels available. Running BB on these labeled datasets can enable us to observe and collect how strong branching computes the branching gains of all candidate variables, selects the best one, and branches upwards and downloads. The branching variables and gains of BB branchings on labeled datasets can be used to identify the key statistical characteristics of the best branching variables that make BB effective and efficient and learn a new branching variable selector.

(i) Data Collection. At every branch in BB, the branching variable selected from multiple candidates can be represented by a binary vector $h = \{0, ..., 0, 1, 0, ..., 0\}$, where 1 indicates "selected" and 0 indicates "not selected". Later, for each candidate branching variable, we extract the features that are related to branching gain as follows:

- Objective function coefficients: Values of objective function coefficients
- Number constraints: Number of constraints that the candidate branching variable participates in.
- Statistics for constraint degree: The degree of a constraint is the number of variables that participate in it. A candidate branching variable may participate in multiple constraints. We use the mean, standard deviation, min, max of the constraint degrees as features.
- Statistics for constraint coefficients: For each candidate branching variable, we can extract a set of coefficients in the constraints the variable participates in. We will use the count, mean, standard deviation, min, max of the set of coefficients as features.
- Statistics for infeasibility: For each candidate branching variable, we count the frequency of selecting the variable that leads to infeasible solutions of subbranches.

Each candidate branching variable, along with its branching gain related features and corresponding binary section label in BB, will be compiled as a training dataset for learning a new branching variable selector. As shown in Figure 1, by incorporating the aggregated listwise branching variable selector into BB, we can mimic the patterns regarding how the strong branching selects the best variables on meta training data by the ranker, and exploit the ranker to enhance our integer programming based recommendations.

(ii) Offline Learning. Next, we construct an aggregated listwise ranking objective to mimic how the Strong Branching strategy selects the best variables for branching. First of all, a BB generates many branches, each of which can be regarded as a ranking task, where the branching variable selector produces a ranking list of candidate branching variables in terms of the features of these candidate variables. Since we can observe many branches, we formulate the modeling of all branches as an aggregated listwise learning to rank problem. From the perspective of generative modeling, we can assume our ranker generates the ranking lists of candidate branching variables in each branch with certain likelihood. Then, learning the branching variable selector is reduced to

the problem of maximizing the likelihood of generating the ranking lists in each branch given our branching variable selector. Formally, suppose there are M branches (i.e., ranking tasks). The m-th branch has N_m candidate branching variables. The training data of m-th branch is denoted by $B_m = \{v_{mn}, h_{mn}\}_{n=1}^{N}$, where v_{mn} and h_{mn} respectively represent the branching gain feature vector and the binary selection label of the n-th candidate branching variable in the m-th branch. Given a candidate variable v_{mn}, we linearly combine its branching gain-related features to predict its ranking score to be selected as a branching variable: $w * v_{mn}$, where w is vectorized linear coefficients. Then, we use softmax functions and listwise strategy to model the likelihood of the predicted ranking list in the m-th branch as:

$$\mathcal{L}(B_m|w) = \prod_{n=1}^{N_m} \left(\frac{e^{wv_{mn}}}{\sum_{n'=1}^{N_m} e^{wv_{mn'}}}\right)^{h_{mn}}. \tag{6}$$

The overall likelihood of all the ranking lists of the M branches is

$$\mathcal{L}(B|w) = \prod_{m=1}^{M} \prod_{n=1}^{N_m} \left(\frac{e^{wv_{mn}}}{\sum_{n'=1}^{N_m} e^{wv_{mn'}}}\right)^{h_{mn}} \tag{7}$$

where $B = \{B_m\}_{m=1}^{M}$. With the formulated overall likelihood, the objective is to find the best estimation of w that maximize the overall ranking likelihood. By applying Jensenfis inequality, the logarithm of $\mathcal{L}$ can be approximated by a lower bound $\mathbb{L}$, which is

$$\begin{aligned} log\mathcal{L}(B|w) &= \sum_{m=1}^{M} \sum_{n=1}^{N_m} h_{mn}(wv_{mn} - log \sum_{n'=1}^{N_m} e^{wv_{mn'}}) \\ &\geq \mathbb{L} = \sum_{m=1}^{M} \sum_{n=1}^{N_m} h_{mn}(wv_{mn} - \sum_{n'=1}^{N_m} wv_{mn'}) \end{aligned} \tag{8}$$

After that, we apply a gradient descent method to maximize $\mathbb{L}$ via $w^{\#it+1} = w^{\#it} + \epsilon \frac{\partial(\mathbb{L})}{\partial w}$, where $\#it$ is iteration number.

(iii) Online Selection. After the branching variable selector is learned, the Strong Branching in BB will be replaced by our selector. Given a set of candidate variables for branching, we use the ranker to rank these candidate variables and branch on the top-1 variable upwards and downwards.

3.3 Recommending Lodging Places

Our proposed system accepts the number of recommended lodging places K, maximum price C, and length of stay T as inputs, which are specified based on a user's preference. After solving the objective, we find the optimal binary vector $\mathbf{y}$ where all the lodging place with indicator 1 will be selected as recommended lodging places. In other words, our proposed system will output a K-sized set of lodging places for the user to explore.

4 EXPERIMENTAL RESULTS

We evaluate the proposed model with real-world data in terms of various tasks.

4.1 Data Description

Table 3 shows the detailed statistics of our real-world data sets for experiments. The data of the lodging places are collected from Airbnb which is a popular website for people to find and rent lodging. It includes the geolocation, neighborhood group (like Manhattan, Brooklyn), room types (private room, entire home), prices, available period and people's ratings and reviews. To estimate the accommodation benefit of the lodging places, we crawled the check-in and POI data from www.foursquare.com. It consists of massive check-in events. Each event is associated with a GPS address, a date, a POI name, a POI category, and comments. For the public transportation, we use the subway data from MTA, which is in the General Transit Feed Specification (GTFS) format, including the information of stops, routes, stop times. To estimate the community risk of the lodging places, we consider 311 complaints data. In New York, 311 is a special telephone number to help local residents to access goverment services and information. The data are composed of a large set of call-in events, each of which are recorded with the complaint type (e.g., Illegal Parking, Noise, Safety), geolocation, location type (e.g., residential, club, restaurant) as well as zip code and created date. The NYPD crime data are also utilized to measure the safety of community. The records in the data are major felony incidents came with timestamp, geolocation and offense type.

Table 3: Statistics of the experiment data.

Data Sources	Properties	Statistics
Airbnb	Number of places	27469
	Size of bounding box (km)	44.28 x 43.175
	Time period of transactions	Jun. 2015 - Sep. 2015
POIs	Number of POIs	12677339
	Number of categories	7
Check-Ins	Number of check-in POIs	12677339
	Number of check-in events	1146283
	Number of POI categories	7
	Time Period	329 days
Crimes	Number of offense types	7
	Number of criminal events	76755
	Number of offense levels	4
	Time Period	36769 days
311 Complaints	Number of location types	137
	Number of complaint types	245
	Number of complaints	9137092
	Time period	4433 days

4.2 Evaluation Metrics

To show the effectiveness of our proposed method, we use Precision, Recall, F-measure, NDCG as evaluation metrics. We evaluate the performance of our proposed method from three aspects: satisfaction ratings, time-on-market (TOM), and review counts. Generally, if a lodging place is good and popular, it is likely that the satisfaction rating of the place is high, the time-on-market of the place is short, and the review count of this place is large.

Precision and Recall. Precision and recall are typically used in a binary rating system. To prepare binary ratings of satisfaction , we collected the original satisfaction scores (ranging from 5 to 1) of each lodging place, and then labeled these lodging places with rating 1 if their satisfaction scores > 4.5, representing the rank of top 10% in population, and rating 0 otherwise. To prepare binary ratings of time-on-market, we collected the number of available days of each lodging place, normalized it by #non-available days / (#available days + #non-available days), and labeled these lodging places with rating 1 if availability ratio > 0.35, representing the rank of top 10% in population, and 0 otherwise. To prepare binary ratings of review counts, we collected the review counts of each lodging place, and then labeled these lodging places with rating 1 if review count > 20, representing the rank of top 10% in population, and 0 otherwise. Thereby, we treat the rating 1 as "relevant" and the rating 0 as

"not relevant". Given a set of N recommended lodging places E_N, precision and recall are defined as Precision@$N = \frac{|E_N \bigcap E_{\geq 1}|}{N}$ and Recall@$N = \frac{|E_N \bigcap E_{\geq 1}|}{|E_{\geq 1}|}$, where $E_{\geq 1}$ are the lodging places whose ratings are greater or equal to 3.

F-measure. F-measure combines precision and recall, and is the harmonic mean of precision and recall. Here we use the F_β measure with $\beta = 0.5$, $xF_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \text{Precision} + \text{Recall}}$ The F_β measure with $\beta < 1$ indicates more emphasis on precision than recall.

Normalized Discounted Cumulative Gain (NDCG). The discounted cumulative gain (DCG) metric is evaluated over top N lodging places on the ranked accommodation list by assuming that better lodging places should appear earlier in the ranked list. The discounted cumulative gain (DCG@N) is given by

$$DCG[n] = \begin{cases} rel_1 & if\ n = 1 \\ DCG[n-1] + \frac{rel_n}{log_2 n}, & if\ n >= 2 \end{cases} \quad (9)$$

where rel_n is the rating of the lodging place n. Later, given the ideal discounted cumulative gain $DCG^{'}$, NDCG at the n-th position can be computed as NDCG[n]= $\frac{DCG[n]}{DCG^{'}[n]}$. The larger NDCG@N is, the higher top-N ranking accuracy is.

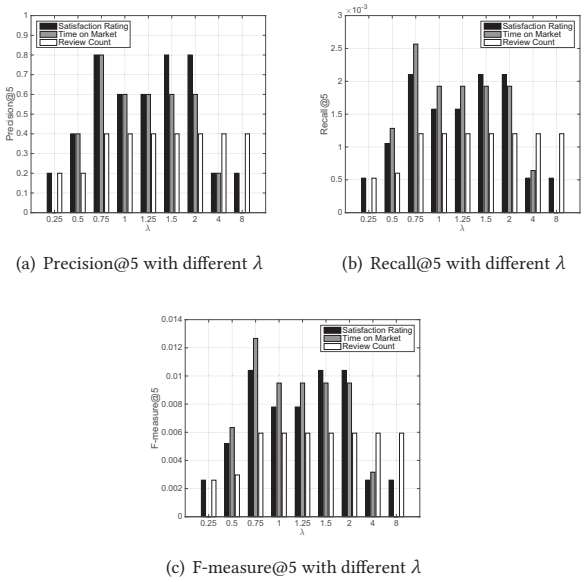
4.3 Baseline Algorithms

Our proposed system accepts the number of recommended lodging places K, maximum price C, and length of stay T, specified by a user, as inputs. Our proposed system will output a K-sized set of lodging places. We thus introduce the following baseline methods for comparison. We name our method as (1) **L-Rec**: we consider both accommodation benefits and community risks, which is exactly our proposed method.

First of all, we compare our method with unsupervised cheaper-first methods. (2) **Price based Ranking (PR) :** we first filter out candidate lodging places based on the constraints of price and length of stay. Then, we rank and select top K cheapest lodging places. In addition, we compare our proposed method with supervised learning-to-rank methods. We then feed a set of lodging places, each of which with extracted features of benefit and risk, as well as corresponding benchmark ratings of satisfaction, time-on-market, and review counts in Airbnb, into the ranking methods for training. After that, we similarly filter out candidate lodging places in terms of price and length of stay, and use the trained models to rank candidate lodging places and select top-K candidates. Specifically, we select three ranking methods: (3) **AdaRank [17]:** AdaRank is a list-wise learning algorithm within the framework of boosting, which can minimize a list-wise loss function. We set tolerance = 0.002 and max consecutive selection count = 5 for AdaRank. (4) **Coordinate Ascent [13]:** it uses domination loss and applies coordinate descent for optimization. We set step base = 0.05, step scale = 2.0, tolerance = 0.001, and slack = 0.001 for Coordinate Ascent . (5) **RankBoost (RB) [8]:** it is a boosted pairwise ranking method, which trains multiple weak rankers and combines their outputs as final rankings. We set the number of iteration = 300, the number of threshold candidates = 10 for RankBoost.

4.4 Parameter Sensitivity of λ

In our objective function, λ controls the weight ratio of accommodation benefits to community risks. We study the sensitivity of λ and identify the best λ. Specifically, we extract the Precision at 5 and Recall at 5 in terms of the benchmark satisfaction ratings, time on market, and review counts, when given different values of λ. Figure 2 shows when λ is set as 0.75, the recommender system achieves the best performances in terms of the three evaluation criteria: satisfaction ratings, time on market, and review counts. A potential interpretation of this finding is that when travelers choose a lodging, they care more about the community risks than the accommodation benefits, since they are traveling to an unfamiliar city and the lodgings on Airbnb are non-standardized hotels. In the next, we will use $\lambda = 0.75$ as a default setting.



(a) Precision@5 with different λ (b) Recall@5 with different λ

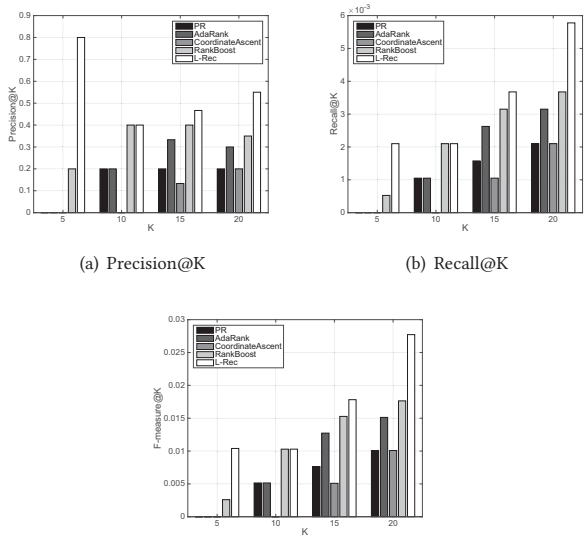
(c) F-measure@5 with different λ

Figure 2: Precision and Recall on satisfaction rating, time on market, and review count with different λ.

4.5 Overall Performances

We compare our method with the baseline methods in terms of Precision and Recall with respect to the three criteria (i.e., satisfaction, time on market, and review counts). Specifically, satisfaction refers to the ratings given by travelers after their stays in the lodging places; time on market refers to the number of days a lodging listing is active on the market, which is an important measurement of popularity and liquidity. The three criteria is used as benchmark ratings (golden standards) for both training and testing in the baseline algorithms. However, the three criteria are only used for testing in our method since our method is unsupervised.

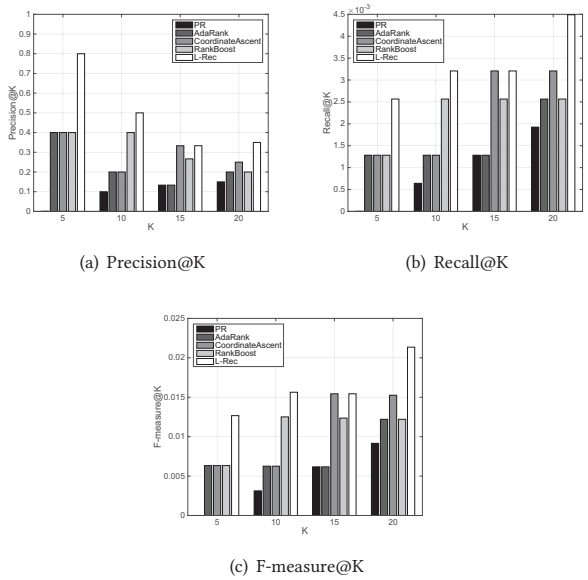
Evaluation on Satisfaction Ratings. We compare our proposed method with the baseline methods in terms of satisfaction ratings. A method is effective if the satisfaction ratings of lodging places recommended by the method are high. In the experiments, we set N as 5, 10, 15, and 20. Figure 3 shows the results of each approach with respect to the three different evaluation metrics. In this figure, we can see that our proposed unsupervised method consistently outperforms not just the price-based ranking method but



(a) Precision@K (b) Recall@K

(c) F-measure@K

Figure 3: Comparison of different methods in terms of precision and recall on satisfaction rating.



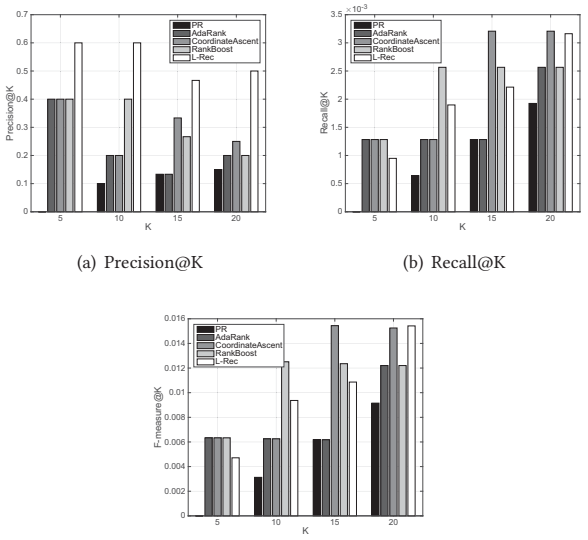
(a) Precision@K (b) Recall@K

(c) F-measure@K

Figure 4: Comparison of different methods in terms of precision and recall on time-on-market.

also the three supervised learning-to-rank methods. The improvement of our unsupervised method is more significant for smaller K, for example, recommending top-5 lodging places.

Evaluation on Time-On-Market. We compare our proposed method with the baseline methods in terms of time on market. A method is effective if the time on market of lodging places recommended by the method are short, which means that the lodgings are quickly rented once they are listed in the rental market. Figure



(a) Precision@K (b) Recall@K

(c) F-measure@K

Figure 5: Comparison of different methods in terms of precision and recall on review count.

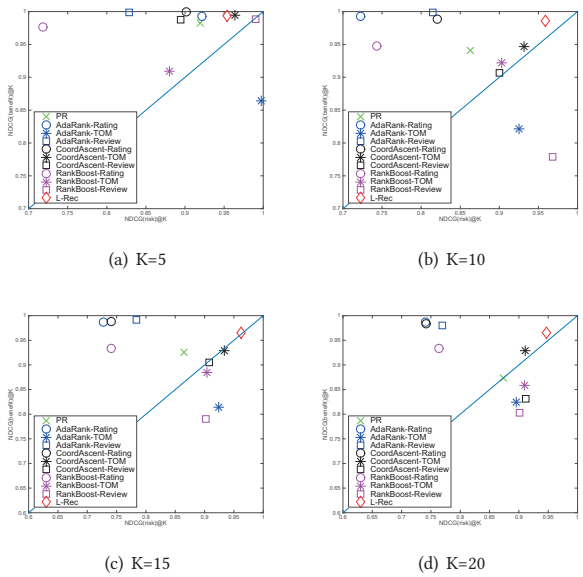
4 shows that when we look at the performances of predicting time-on-market, our proposed method again consistently outperform the baseline methods, particularly for hitting top-5 and top-20 lodging places. Also, the two supervised methods, AdaRank and RankBoost, are in the second place. The price based ranking again achieves lowest accuracy.

Evaluation on Review Count. We compare our proposed method with baseline methods in terms of review count. A method is effective if the review numbers of lodging places recommended by the method are large. Figure 5 shows that the performance of our proposed method ranks #1 with respect to precision and ranks #3 in terms of recall. Although Coordinate Ascent and RankBoost methods outperform our method with respect to recall and f-measure, our method still outperforms price based ranking (unsupervised), as well as AdaRank (supervised), which is traditionally expected to outperform our unsupervised method.

Look back to all the three criteria (i.e. satisfaction rating, time-on-market, review count), our method stand out from all the compared methods due to the following aspects: (i) our proposed method strive a balanced performance among the three criteria; (2) our proposed method is unsupervised without expensive quality ratings as supervised information.

4.6 Balance Check between Benefit and Risk

We evaluate our recommendation approach by checking whether it can achieve a balance between accommodation benefits and community risks. Specifically, there are two measures of lodging places, i.e., accommodation benefit scores and community risk scores. If a recommendation system outputs a list of top-K lodging places, we use (i) a benefit emphasis metric, measured by NDCG(benefit) and (ii) a risk emphasis metric, measured by NDCG(risk), to check the balance of a top-K list between benefit emphasis and risk emphasis. To compute the NDCG(benefit) and NDCG(risk) of a top-K



(a) K=5 (b) K=10

(c) K=15 (d) K=20

Figure 6: Balance check on different methods between NDCG(benefit)@K and NDCG(risk)@K. AdaRank-Rating: AdaRank trained on satisfaction ratings; AdaRank-TOM: AdaRank trained on time on market; AdaRank-Review: AdaRank trained on review count. The naming policy is similar for other methods.

list, the relevance scores *rel* in Equation 9 are set to be the benefit score and the risk score, respectively. Intuitively, if a top-K list has higher NDCG(benefit) or NDCG(risk), it means that the list has more emphasis on the positive or negative side. Thereby, we prefer a top-K list with both high NDCG(benefit) and high NDCG(risk). Figure 6 visualizes the balance check of different methods on top-K (K=5, 10, 15, 20) recommendation between benefit emphasis and risk emphasis, where x-axis represents risk emphasis and y-axis represents benefit emphasis. We can observe that our proposed method is located in the upper-right corners of four subfigures. This observation indicates our method has both highest benefit emphasis and highest risk emphasis. Also, with the increase of K, the recommendation results of both our method and the baselines have more emphasis on risks than benefits.

4.7 Efficiency Study

We compare our enhanced learning to integer programming (LIP) with three baseline methods: (i) traditional branch and bound with strong branching (BB) [2] ; (ii) simulated annealing (SA) [3] ; and (iii) genetic algorithm (GA) [4] . While GA and SA are developed for local optimum, BB and our enhanced BB are developed for global optimum. Theoretically, BB and our LIP method can outperform GA and SA with respect to recommendation effectiveness and robustness. Here, we focus on comparing their running time in terms of different percentages of candidate lodging places. Table 4 shows our LIP method can greatly reduce the time cost compared with BB, GA, and SA. The reasons are: GA and SA are probabilistic iterative searching methods and their convergence requires more iterations. Compared with BB, LIP combines both BB and a branching variable selector learned via an aggregated listwise ranking strategy. The selector can mimic and learn how to select the best branching variable. Therefore, LIP method can save time.

[2] https://cran.r-project.org/web/packages/lpSolve/
[3] https://cran.r-project.org/web/packages/GenSA/
[4] https://cran.r-project.org/web/packages/genalg/

Table 4: Running time comparison (in seconds) between our method (LIP) and baseline methods over different percentages of training data.

Percentage	GA	SA	BB	LIP
0.4	663.645	985.657	13.121	**19.134**
0.6	689.848	1064.740	44.873	**30.419**
0.8	759.948	1152.539	152.189	**82.872**
1	867.753	1208.099	235.459	**122.287**

4.8 Discussions

Different from traditional supervised methods (e.g., collaborative filtering, matrix factorization), we focus on an alternative and different perspective by formulating recommendation in P2P environments as an unsupervised problem based on integer programming, where utility of recommendations is maximized by considering positive and negative sides while personalizations are implemented based on constraints. We find that such an unsupervised strategy can be very helpful for simultaneously achieving a balance among multiple criteria (satisfaction, time on market, number of reviews). Also, we adapt the idea of learning by analogy and combine the branch and bound method with a branching variable selector, which is learned by listwise learning to rank. The strategy can improve the efficiency. From the preliminary results, we believe it is promising to exploit more supervised learning methods to learn and mimic the decision making processes of searching optimal solutions in integer programming.

5 RELATED WORK

The related work can be categorized into the following categories: (i) utility-based recommendations; (ii) learning to optimize, (vi) P2P rental markets.

Utility based Recommendations. Our work is relevant to utility based recommendations. Most recommendation approaches utilize users' ratings for items to predict their preferences, and recommend users with items that have highest predicted ratings. Nevertheless, there also exist several work that ignore the ratings due to their low quality and sparsity, and different optimization goals. For instance, users' utility functions are defined to better capture multiple dimensions of recommender systems, since rational users aim to maximize their utility in the decision process [14]. [6] further decomposed the overall product utility as the sum-product of contributions of interest dimensions and the utility in each dimension. [15] incorporated explicit utility into recommendation process, which is revealed by users' actions. A mixed integer programming (MIP) is formulated by introducing the setwise minmax decision rules, and items are recommended by solving the optimization problem. [4] designed a utility function for query search, and recommend queries by optimizing the utility function. [16] defined

a marginal net utility function to better match users' purchase decisions in e-commerce websites, and further designed a utility based SVD algorithm to recommend items.

Learning To Optimize. Our work is related to the research of Learning To Optimize that applies machine learning approaches to improve optimization. The idea of learning heuristic functions for combinatorial search algorithms dates back to [5, 12]. Particularly, [5] proposed a learning approach named STAGE to automatically improving search performance on optimization problems. Recently, [2] developed a set of features to describe the states of ongoing optimization procedures and used extremely randomized trees to predict strong branching scores. [3] proposed a machine learning approach to load balancing in parallel branch-and-bound for MIP. [9] presented a method for learning an adaptive node searching order for any class of problem solvable by branch-and-bound. More recently,[11] collected and combined the strong branching scores of all the branching variables during BB into a single global ranking list, and trained a previously-proposed pairwise ranking function for selecting variables. However, in many cases, there is no global metric, i.e., strong branching scores, for compiling a global ranking list of candidate branching variables.

To be generalized, we use binary indicator (selected or not-selected) to represent the local rankings at each branch during BB, and thus there are multiple local ranking lists for the entire BB process. Our proposed aggregated listwise ranking method can aggregate these local ranking lists and collectively learn the patterns about how BB selects the best branching variables. Therefore, our method is more robust and generalized.

P2P Rental Markets. Finally, our work has a connection with P2P rental markets. With regards to the application of P2P rental markets, recent work focused on the market efficiency, searching and ranking problems on the market. [7] studied the efficiency of P2P markets and the effects of ranking algorithms on the search and matching process using internal data from Airbnb. The results show that potential guests engaged in limited search are frequently rejected by hosts, and match at lower rates, while an algorithm with personalized constraints would increase the matching rate by up to 10%. [18] revealed the nuances of user-generated ratings in the context of the sharing economy and find that there only exists weak correlations between the ratings of the listings across different platforms. Our work aims to resolve one of the challenges in P2P rental markets, that is, to address rating sparsity and quality issues, provide effective recommendations, and overcome information imbalance between providers and consumers.

6 CONCLUSION REMARKS

Traditional single rating-based recommender systems (e.g., collaborative filtering) may often encounter challenges when ratings are sparse, dynamic, and expensive. According to Multi-Attribute Utility Theory, choices are made in terms of multiple criteria, which inspire people to seek for information from heterogeneous data sources, and further make choices by navigating the multiple criteria and maximizing utility. Following the decision process, we have presented an unsupervised optimization framework to effectively and efficiently recommend products with the goal of maximizing the utility under multi-criteria concerns. Particularly, we apply the framework to the P2P lodging rental markets.

To accomplish the goal, we firstly propose a fast unsupervised fused scoring method for assessing both the benefits and risks of products from cross-domain data, in order to objectively evaluate the utility of products. Besides, the objective of maximizing the utility is closely related to a knapsack problem, and thus a constrained integer programming problem is formulated to provide flexibility for navigating multiple criteria. In addition, we propose an enhanced branch-and-bound method by incorporating aggregated listwise ranking for enhancing efficiency. The aggregated listwise ranker can mimic and learn the decision policy of strong branching and speed up the selection of branching variables. Finally, the empirical experiments on P2P lodging rental market data has demonstrated the effectiveness in striking a trade-off among user satisfaction, time on market, and number of reviews, while the optimization efficiency is significantly improved.

Acknowledgment. This is research was partially supported University of Missouri Research Board (UMRB) via proposal number: 4991.

REFERENCES

[1] Tobias Achterberg, Thorsten Koch, and Alexander Martin. 2005. Branching rules revisited. *Operations Research Letters* 33, 1 (2005), 42–54.
[2] Alejandro Marcos Alvarez, Quentin Louveaux, and Louis Wehenkel. 2014. A supervised machine learning approach to variable branching in branch-and-bound. In *In ECML*. Citeseer.
[3] Alejandro Marcos Alvarez, Louis Wehenkel, and Quentin Louveaux. 2014. Machine Learning to Balance the Load in Parallel Branch-and-Bound. (2014).
[4] Aris Anagnostopoulos, Luca Becchetti, Carlos Castillo, and Aristides Gionis. 2010. An optimization framework for query recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 161–170.
[5] Justin A Boyan and Andrew W Moore. 1998. Learning evaluation functions for global optimization and boolean satisfiability. In *AAAI/IAAI*. 3–10.
[6] Alexander Felfernig, Bartosz Gula, Gerhard Leitner, Marco Maier, Rudolf Melcher, and Erich Teppan. 2008. Persuasion in knowledge-based recommendation. In *International Conference on Persuasive Technology*. Springer, 71–82.
[7] Andrey Fradkin. 2013. *Search frictions and the design of online marketplaces*. Technical Report. Tech. rep., Working paper, Stanford University.
[8] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *The Journal of machine learning research* (2003).
[9] He He, Hal Daume III, and Jason M Eisner. 2014. Learning to Search in Branch and Bound Algorithms. In *Advances in Neural Information Processing Systems*. 3293–3301.
[10] Xiaofei He, Deng Cai, and Partha Niyogi. 2005. Laplacian score for feature selection. In *Advances in neural information processing systems*. 507–514.
[11] Elias B Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. 2016. Learning to Branch in Mixed Integer Programming. In *Thirtieth AAAI Conference on Artificial Intelligence*.
[12] Matthew Lowrie and Benjamin Wah. 1988. Learning heuristic functions for numeric optimization problems. In *Computer Software and Applications Conference, 1988. COMPSAC 88. Proceedings., Twelfth International*. IEEE, 443–450.
[13] Donald Metzler and W Bruce Croft. 2007. Linear feature-based models for information retrieval. *Information Retrieval* (2007).
[14] J von Neumann, Oskar Morgenstern, and others. 1944. *Theory of games and economic behavior*. Vol. 60. Princeton university press Princeton.
[15] Paolo Viappiani and Craig Boutilier. 2009. Regret-based optimal recommendation sets in conversational recommender systems. In *Proceedings of the third ACM conference on Recommender systems*. ACM, 101–108.
[16] Jian Wang and Yi Zhang. 2011. Utilizing marginal net utility for recommendation in e-commerce. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 1003–1012.
[17] Jun Xu and Hang Li. 2007. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 391–398.
[18] Georgios Zervas, Davide Proserpio, and John Byers. 2015. A First Look at Online Reputation on Airbnb, Where Every Stay is Above Average. *Where Every Stay is Above Average (January 23, 2015)* (2015).