Mobile Crowdsensing Task Recommendation Based on Spatiotemporal Information and Task Popularity Analysis: Postprint

Authors: Yang Guisong, Wang Jingru, Li Jun, He Xingyu

Date: 2022-05-10T11:22:58Z

Abstract

Existing mobile crowdsensing task recommendation methods share common drawbacks: on one hand, they fail to fully consider the impact of spatio-temporal information on worker preferences, resulting in low recommendation accuracy; on the other hand, they ignore the influence of task popularity on recommendations, leading to poor recommendation coverage. To address these issues, this paper proposes a mobile crowdsensing task recommendation method based on spatio-temporal information and task popularity analysis. First, we fully utilize relevant information from worker execution records (such as the time and location when workers perform tasks) to accurately predict worker preferences for tasks. Second, we analyze task popularity based on worker reputation and task execution status, and design a task popularity penalty factor to improve the coverage of recommendation results. Then, we generate a task recommendation list by combining worker preferences and the popularity penalty factor. Experimental results demonstrate that, compared with existing baseline methods, the proposed method achieves an average improvement of 3.5% in recommendation accuracy and 25% in recommendation coverage.

Full Text

Preamble

Vol. 39 No. 9
Application Research of Computers (ChinaXiv Cooperative Journal)

Task Recommendation Based on Spatial-Temporal Information and Task Popularity Analysis in Mobile Crowd Sensing

Yang Guisong[1] , Wang Jingru[1] , Li Jun[2], He Xingyu[1] †
(1. University of Shanghai for Science & Technology, a. School of Optical-

Electrical & Computer Engineering; b. College of Communication & Art Design, Shanghai 200093, China;
2. National Industrial Information Security Development Research Center, Beijing 100040, China)

Abstract: Existing task recommendation methods in mobile crowd sensing suffer from two common drawbacks: first, they fail to adequately consider the influence of spatial-temporal information on worker preferences, resulting in low recommendation accuracy; second, they ignore the impact of task popularity on recommendations, leading to poor coverage. To address these issues, this paper proposes a novel task recommendation approach based on spatial-temporal information and task popularity analysis. The method first leverages relevant information from worker execution records (e.g., task execution time and location) to accurately predict worker preferences. Second, it analyzes task popularity based on worker reputation and task execution patterns, designing an appropriate task popularity penalty factor to improve recommendation coverage. Finally, it generates task recommendation lists by combining worker preferences with the popularity penalty factor. Experimental results demonstrate that compared with existing baseline methods, the proposed approach improves recommendation accuracy by 3.5% on average and increases recommendation coverage by 25%.

Keywords: mobile crowd sensing; task recommendation; spatial-temporal information; popularity bias; task popularity

0 Introduction

Mobile Crowd Sensing (MCS) represents a promising sensing paradigm that has been widely applied across numerous domains, including environmental monitoring, health data collection, industrial control management, geological surveying, and transportation planning. A typical MCS system comprises three components: task publishers, task workers, and an MCS platform. The workflow proceeds as follows: task publishers submit requests to the MCS platform, which establishes task lists for workers based on received requests; workers select tasks from these lists to execute, upload sensed data to the platform, which then delivers the data back to publishers. Throughout this process, task selection likelihood is influenced by worker preferences. Consequently, mining worker preferences to recommend the most probable task lists becomes crucial. The efficiency and quality of task recommendations not only affect workers' willingness to participate but also impact the quality of sensed data.

Existing research tends to employ single models to learn worker preferences for task recommendation. For instance, methods based on worker similarity or task similarity, and approaches using logistic regression models, require predefining factors that influence worker preferences. However, in MCS, numerous factors affect preferences, making it impractical to predetermine all of them. To address this limitation, some studies have begun applying recommender systems to MCS task recommendation, predicting worker preferences from historical execution records. For example, certain works utilize matrix factorization models to predict preferences and generate recommendation lists, while others propose feature-based Bayesian methods to learn latent task features and address cold-start scenarios for new tasks. Some approaches integrate both worker preferences and reliability for task recommendation.

These methods inadequately consider the spatial-temporal dynamics of worker preferences, which affects recommendation accuracy. On one hand, preferences evolve over time, with recent task selections better representing current interests. On the other hand, MCS tasks are distributed across different sensing regions, requiring workers to travel to execution locations. Considering cost factors, workers prefer tasks nearer to their current location. Although one study employs tensor decomposition for location-based recommendation, worker mobility necessitates frequent model updates to adapt to location changes, preventing timely capture of new preferences. Therefore, accurately predicting worker preferences by incorporating spatial-temporal information remains a significant challenge.

Furthermore, existing MCS task recommendation research overlooks the impact of task popularity on recommendation effectiveness, leading to popularity bias and reduced coverage. For example, popular tasks like urban traffic monitoring may attract many workers due to their simplicity. If the platform recommends such tasks to all workers, it creates data redundancy, while unpopular tasks like mountainous geological surveys may fail due to insufficient samples. Typically, in recommender systems, task popularity is determined by the proportion of workers executing it—the higher the proportion, the greater the popularity. In MCS platforms, workers exhibit herd mentality, easily influenced by others to execute certain tasks, causing popularity fluctuations. Task popularity is closely related to worker reputation, as higher-reputation workers exert greater influence on others, consequently impacting task popularity more significantly. Thus, analyzing task popularity based on worker reputation and execution patterns, and designing reasonable popularity penalty factors to improve coverage, presents another challenge.

To address these challenges, this paper proposes the TimeMF-BiLSTM method (Time Matrix Factorization and Bidirectional Long Short-Term Memory) for MCS task recommendation. This approach comprehensively considers worker preferences and task popularity penalty factors to generate recommendation lists, ensuring both accuracy and coverage. The main contributions are:

a) A time-aware matrix factorization model learns implicit relationships between workers and tasks from historical execution data, while a BiLSTM model extracts location preference information from worker trajectory data. Fusing these two models yields more accurate task recommendations.

b) Social networks are introduced to calculate worker reputation, and a task popularity calculation method based on worker reputation is proposed for more precise popularity analysis. A reasonable task popularity penalty factor is designed to ensure recommendation coverage.

1 System Model

Figure 1 illustrates the overall architecture of the TimeMF-BiLSTM task recommendation method. The approach first employs a time-aware matrix factorization model (TimeMF) to learn implicit relationships between workers and tasks (i.e., latent worker feature vectors and task latent features) from execution records, deriving worker preferences (denoted as TimeMF preference scores) to address the dynamic temporal changes in preferences. Second, a Bidirectional Long Short-Term Memory (BiLSTM) network learns workers' likely next locations from GPS trajectory sequences, generating worker preferences based on distances between tasks and that location (denoted as BiLSTM preference scores) to address dynamic spatial changes in preferences. Third, the preferences learned by TimeMF and BiLSTM are fused using logistic regression to obtain comprehensive worker preference scores, ensuring recommendation accuracy. Additionally, social networks are introduced to calculate worker reputation, and influence factors among workers are analyzed based on task execution patterns. Combining worker reputation and influence factors enables task popularity calculation, constructing a penalty factor negatively correlated with popularity. The final worker-task score is obtained by multiplying the popularity penalty factor with worker preferences, addressing coverage imbalance caused by popularity bias. Finally, tasks are ranked by these final scores in descending order to generate Top-N recommendation lists for each worker.

Key parameters are described as follows: the worker set contains m workers $\{u_1, u_2, ..., u_m\}$, the task set contains n tasks $\{v_1, v_2, ..., v_n\}$, worker execution records are represented as $\mathbb{H}$, containing all tasks executed by worker u_i, each task's execution record is represented as $\mathbb{E}$, including all worker IDs who executed task v_j, and $\mathbb{L}$ represents worker u_i' s GPS trajectory sequence where l_s is a latitude-longitude pair denoting worker u_i' s location at time t_s. This paper proposes an effective method to accurately compute the worker-task scoring matrix Y, generating Top-N recommendation lists for each worker with the goal of maximizing recommendation accuracy and coverage, where N is set by the platform based on actual conditions.

2.1 Time-Aware Matrix Factorization

To address dynamic temporal changes in worker preferences, the TimeMF model learns preferences through iterative updates of latent worker feature vectors U

and task latent feature vectors V, using their product to predict the worker-task scoring matrix $\hat{MP}$ and fill in zero values in the original matrix MP. TimeMF comprises three components: (a) initial worker-task scoring matrix construction based on execution records with temporal factors; (b) model learning using appropriate algorithms to iteratively update task latent features until reaching an optimal solution where the product of worker and task latent feature vectors approximates the original matrix; (c) preference score generation based on the final latent vectors to predict worker-task scores.

2.1.1 Initial Worker-Task Scoring Matrix Construction

Worker-task scores represent preferences. Since MCS platforms rarely require explicit ratings, this paper uses task execution frequency as implicit ratings, represented as a triple $\{u_i, v_j, t_{ij}\}$ where t_{ij} is the execution time. If worker u_i executes task v_j at time t_{ij}, it indicates interest $\{u_i, v_j, t_{ij}\} = 1$; otherwise $\{u_i, v_j, t_{ij}\} = 0$.

Considering the temporal dynamics of preferences, recent tasks better represent current interests. Therefore, inspired by Newton's law of cooling, an exponential decay function $e^{-\lambda(t-t_{ij})}$ is introduced, making score weights decay exponentially over time, where t is the current time, t_{ij} is the execution time, and $\lambda > 0$ is the temporal decay factor (larger λ means lower importance of historical preferences). The original worker-task scoring matrix MP is constructed where each element mp_{ij} is calculated as:

$$mp_{ij} = \sum_{\{t_{ij} | (u_i, t_{ij}) \in \mathbb{H}\}} e^{-\lambda(t-t_{ij})}$$

2.1.2 TimeMF Model Learning

Worker and task latent feature vectors are represented by two k-dimensional low-rank matrices $U \in \mathbb{R}^{m \times k}$ and $V \in \mathbb{R}^{n \times k}$, where each row of U represents worker u_i' s latent vector U_i and each row of V represents task v_j' s latent vector V_j. Values are randomly initialized. The training objective minimizes the error between $\hat{MP}$ and the original matrix MP:

$$\arg\min_{U,V} \frac{1}{2} ||MP - UV^T||_F^2 + \frac{\lambda_U}{2} ||U||_F^2 + \frac{\lambda_V}{2} ||V||_F^2$$

where $c_{i,j}$ indicates worker u_i' s interest level in task v_j, defined as $c_{i,j} = 1 + \alpha \cdot mp_{ij}$ with α as a decay parameter. $|| \cdot ||_F$ denotes the Frobenius norm, and λ_U and λ_V are regularization terms. Equation (2) can be solved using gradient descent methods.

2.1.3 TimeMF Preference Score Generation

Using the final learned latent vectors, the preference score is generated as:

$$\hat{mp}_{ij} = U_i V_j^T$$

2.2 Spatial-Aware Bidirectional Long Short-Term Memory

To address dynamic spatial changes in worker preferences, BiLSTM learns preferences through: (a) trajectory data processing with missing value imputation; (b) BiLSTM prediction of workers' likely next locations from GPS trajectories; (c) preference score generation based on distances between the predicted location and task locations.

2.2.1 Trajectory Data Processing

For missing values in worker trajectory data, imputation first leverages spatio-temporal correlations within the worker's own trajectory. If worker u_i's location at time t_s is missing, the system searches the preceding and following 5 time slots. If location information is found, it fills the missing value l_s.

For remaining missing values, trajectory similarity between worker u_i and other workers is compared to fill gaps using similar workers' location information at that time. If only one similar worker has location data, it is used directly; if multiple workers have data, the most similar worker' s location is selected.

Trajectory similarity between workers u_i and u_w is calculated as:

$$\xi(l_s^{(u_i)}, l_s^{(u_w)}) = \sum_{t_s \in LS} \xi(l_s^{(u_i)}, l_s^{(u_w)})$$

where $\xi(l_s^{(u_i)}, l_s^{(u_w)})$ represents location similarity at time t_s:

$$\xi(l_s^{(u_i)}, l_s^{(u_w)}) = \begin{cases} 1 & \text{if } d(l_s^{(u_i)}, l_s^{(u_w)}) \leq \epsilon \\ 0 & \text{if } d(l_s^{(u_i)}, l_s^{(u_w)}) > \epsilon \end{cases}$$

$d(l_s^{(u_i)}, l_s^{(u_w)})$ denotes the distance between workers at time t_s. If the distance is less than or equal to threshold ϵ, the locations are considered similar. The ϵ value is set by the platform.

2.2.2 BiLSTM Prediction

Since BiLSTM requires fixed-length input sequences, a sliding window method is introduced (window size determined experimentally). Trajectory sequences are converted into fixed-length input-output samples for training.

The BiLSTM model consists of 5 layers: (a) Input layer receiving worker-task trajectory sequences; (b) Embedding layer using Word Embedding to convert GPS sequences into vector representations x_{ID}; (c) BiLSTM layer with forward and backward LSTM hidden layers processing x_{ID} to produce vectors h_1 (forward) and h_2 (backward); (d) Dropout layer to prevent overfitting; (e) Output layer predicting the next likely location l_{next}.

2.2.3 BiLSTM Preference Score Generation

The worker-task preference matrix LP is constructed where each element lp_{ij} is derived using Euclidean distance:

$$lp_{ij} = ||l_{next} - l_j^v||_2$$

l_j^v represents task v_j' s required sensing location. If the predicted next location l_{next} is close to task v_j, the worker is more likely to execute it.

2.3 Preference Fusion

The final worker preference score P is obtained by fusing TimeMF scores $\hat{MP}$ and BiLSTM scores LP using logistic regression:

$$p_{ij} = \frac{1}{1 + \exp(-(\hat{mp}_{ij} + lp_{ij}))}$$

Each element p_{ij} represents worker u_i' s fused preference score for task v_j.

---

3 Task Popularity Penalty Factor Design

Traditional popularity calculation methods treat all workers' impact on task popularity as homogeneous, introducing errors. In MCS platforms, task popularity is closely related to worker reputation. Workers exhibit herd mentality, easily influenced by others. When high-reputation workers execute a task, it attracts more workers in the next time slot, causing popularity spikes. Therefore, incorporating worker reputation enables better prediction of future popularity.

First, the worker influence factor matrix IN is computed, representing how easily workers affect each other. The influence factor in_{iw} between workers u_i and u_w is:

$$in_{iw} = \frac{|\mathbb{H}_i \cap \mathbb{H}_w|}{|\mathbb{H}_i \cup \mathbb{H}_w|}$$

where $\mathbb{H}_i$ and $\mathbb{H}_w$ are task sets executed by each worker. More common tasks indicate higher intimacy and mutual influence.

Second, social networks are introduced to compute worker reputation, establishing reputation matrix WP. Reputation can be quantified through likes, comments, and friend relationships. The impact of worker u_i' s reputation on task v_j' s popularity is $wp_i \cdot in_{iw}$. Thus, task popularity is redefined as:

$$pop_j = \sum_{u_i \in \mathbb{U}_j} \sum_{u_w \in \mathbb{U}} wp_i \cdot in_{iw}$$

where $\mathbb{U}_j$ is the set of workers who executed task v_j.

Task popularity values are normalized to the range [0,1]:

$$pop_j^{norm} = \frac{pop_j - pop_{min}}{pop_{max} - pop_{min}}$$

where pop_{max} and pop_{min} are maximum and minimum popularity values in the task set.

The popularity penalty factor m_j for task v_j adjusts recommendation weights to ensure coverage:

$$m_j = \begin{cases} 1 & \text{if } sample_j < \phi \\ 1 - \frac{sample_j - \phi}{sample_j^{total} - \phi} \cdot pop_j^{norm} & \text{if } \phi \le sample_j < sample_j^{total} \end{cases}$$

$sample_j$ is the current number of collected samples, $sample_j^{total}$ is the required sample count (defined by task publishers), and ϕ is the median execution count of all tasks in the platform.

---

4 Task Recommendation List Generation

Based on the popularity penalty factor, the final worker-task prediction score is:

$$y_{ij} = p_{ij} \cdot m_j$$

where p_{ij} is the fused spatial-temporal preference and m_j is the popularity penalty factor. When $sample_j < sample_j^{total}$, tasks are ranked by y_{ij} in descending order, and the top N tasks form the recommendation list. When $sample_j \geq sample_j^{total}$, the task is removed from the candidate set.

Algorithm 1: TimeMF-BiLSTM

Input: Worker set $\mathbb{U}$, task set $\mathbb{V}$, worker execution records $\mathbb{H}$, task execution records $\mathbb{E}$, worker trajectory records $\mathbb{L}$, required sample counts $sample^{total}$

Output: Worker task recommendation lists

```
a) Randomly initialize worker latent vectors $U$ and task latent vectors $V$
b) Construct original scoring matrix $MP$ using equation (1)
c) Construct loss function from equation (2) and train TimeMF via gradient descent; if loss < threshold, proceed to (d), else repeat (c)
d) Build TimeMF preference matrix $\hat{MP}$ using equation (3)
e) Compute trajectory similarity between workers $u_i$ and $u_w$ using equation (5)
f) For each worker $u_i$, traverse trajectory record $\mathbb{L}$; if $l_s^{(u_i)} = 0$, fill using the most similar worker' s location at that slot
g) Train BiLSTM model to output predicted next location $l_{next}$
h) Compute similarity $lp_{ij}$ between $l_{next}$ and task $v_j$' s location using equation (6), constructing BiLSTM preference matrix $LP$
i) Fuse preferences using equation (7) to obtain final preference matrix $P$
j) Construct popularity penalty factor matrix $M$ using equation (13)
k) Compute final prediction score matrix $Y$ using equation (14)
l) For each worker $u_i$, sort row $i$ of $Y$ in descending order, output top $N$ tasks as recommendation list; if $sample_j \geq sample_j^{total}$ for task $v_j$, remove $v_j$ from $\mathbb{V}$ and restart training
```

5.1 Datasets

The proposed method is validated on two real datasets: Gowalla and Foursquare.

Gowalla Dataset: Contains 196,591 users' check-in records from August 2009 to September 2010. After preprocessing (removing workers with <20 check-ins), 216,245 records remain, including 78 workers, 117 tasks, and 371 locations.

Foursquare Dataset: Contains 2,153,471 users, 1,143,092 venues, 1,021,970 check-ins, 27,098,490 social relations, and 2,809,581 ratings extracted via public API. After preprocessing, 622,841 records remain, including 285 workers, 105 tasks, and 168 locations.

Experiments run on a Dell laptop with Intel Core i5-10210U processor and 8GB RAM, Windows 10 OS, Python 3.6. Main parameters are listed in Table 1.

Table 1. Parameter Settings

Parameter	Value Range
Time slot T	30min
Task count n	[20,100]
Worker count m	[10,25]
Latent feature dimension k	[0.001,0.05]
Learning rate α	[0.01,0.9]
Regularization β	-
Time window t	-
Dropout	-

5.3 Evaluation Metrics

a) Precision

$$Precision = \frac{1}{m}\sum_{i=1}^{m}\frac{hitcount_i}{total\_recommendation\_number_i}$$

where $hitcount_i$ is the number of tasks executed by worker u_i from the recommendation list.

b) Normalized Discounted Cumulative Gain (NDCG)

NDCG measures ranking quality, with values in (0,1). Higher NDCG indicates better alignment with worker preferences.

$$NDCG = \frac{DCG}{IDCG}$$

where DCG is Discounted Cumulative Gain:

$$DCG = \sum_{i=1}^{b}\frac{rel_i}{\log_2(i+1)}$$

b is the number of tasks in the recommendation list, and rel_i is 1 if the worker executed the task, 0 otherwise.

c) Coverage

Coverage describes the proportion of tasks that can be recommended:

$$Coverage = 1 - \frac{1}{n} \sum_{c=1}^{n} \frac{pop(v_c)}{\sum_{j=1}^{n} pop(v_j)}$$

where v_c is the c-th task sorted by popularity in descending order.

5.4 Comparison Methods

The proposed method is compared against six baselines:

- **TR-UMCR**: Collaborative ranking combining hybrid user models with list-wise ranking learning
- **FLTE**: Logistic regression-based task recommendation
- **RTRA**: Matrix factorization only (learns implicit worker-task relationships)
- **PRTR**: Integrates location information using tensor decomposition, ignoring temporal dynamics
- **TCTR**: Considers only temporal factors, ignoring spatial dynamics
- **RLIN**: Linear weighted fusion of two models (vs. logistic regression in TimeMF-BiLSTM)

5.5 Performance Evaluation

Precision: Tables 2 and 3 show precision results on both datasets. TimeMF-BiLSTM significantly outperforms FLTE by leveraging implicit worker-task relationships. When $N > 15$, TimeMF-BiLSTM exceeds TR-UMCR, PRTR, and TCTR, demonstrating that fusing spatial-temporal factors better captures preferences. TR-UMCR and PRTR perform similarly as both use basic matrix factorization with location information, confirming location’ s importance in MCS. Compared to RLIN, TimeMF-BiLSTM achieves slightly higher precision because integer ratings from execution history cause rounding errors in linear regression. Precision increases with N, converging to ~94.3% on Gowalla when $N = 20$.

Table 2. Precision on Gowalla Dataset

Method	N=5	N=10	N=15	N=20
TimeMF-BiLSTM	32.2%	59.6%	88.4%	94.3%
TR-UMCR	36.8%	21.9%	25.2%	39.6%
FLTE	37.7%	30.2%	58.4%	30.7%
RTRA	38.1%	59.9%	60.3%	58.5%
PRTR	82.9%	64.1%	73.8%	78.3%
TCTR	80.8%	82.1%	90.2%	70.5%
RLIN	82.4%	88.9%	87.6%	91.6%

Table 3. Precision on Foursquare Dataset

Method	N=5	N=10	N=15	N=20
TimeMF-BiLSTM	33.9%	57.5%	78.9%	92.1%
TR-UMCR	27.7%	21.0%	25.2%	20.7%
FLTE	37.7%	31.6%	57.1%	25.1%
RTRA	33.9%	43.5%	60.3%	53.4%
PRTR	77.5%	57.1%	70.1%	77.4%
TCTR	84.8%	81.8%	89.2%	63.3%
RLIN	78.5%	89.2%	85.1%	89.5%

When $N < 10$, TimeMF-BiLSTM' s precision is slightly lower than PRTR due to the popularity penalty factor slightly reducing accuracy for coverage improvement. To verify this, worker ID 9 from Foursquare is analyzed using tasks 1-20 as training data. Table 4 shows that predicted preferences P closely match true preferences (MSE = 0.414152). However, without penalty, tasks 2,3,4,8 (high-scoring) would dominate the top-5 recommendations, causing popularity bias. After applying the penalty factor, final scores Y reduce weights for popular tasks, enabling tasks from regions C1 and C2 to rank higher, improving coverage at a slight precision cost. As N increases, coverage requirements are met and precision improves.

Table 4. Sample Instance

Task ID	True Score	Region	Predicted P	Final Y
1	4	C1	3.8	3.8
2	8	C3	7.9	5.2
3	10	C3	9.8	6.1
4	7	C4	6.9	4.8
5	3	C1	2.9	2.9

NDCG: With $N = 20$ and parameter $b = n$ (current task count), TimeMF-BiLSTM achieves significantly higher NDCG than baselines, nearly double that of logistic regression fusion, confirming superior ranking accuracy.

Coverage: Figures 4 and 5 show coverage positively correlates with N. TimeMF-BiLSTM achieves higher coverage by penalizing popular tasks, enabling more diverse task selection. When worker-task ratio increases, coverage decreases proportionally, but the decline is slower with reputation integration (Figures 6-7). At a 10:1 ratio, coverage exceeds 71%. More social relationship entries further slow the coverage decline, validating the importance of worker reputation.

---

6 Conclusion

This paper addresses low accuracy and coverage in MCS task recommendation by proposing a method that fuses spatial-temporal information with task popularity analysis. The approach accurately mines worker preferences while mitigating popularity bias. By separately computing temporal and spatial influences, it reduces the need for frequent model updates. The method is applicable to all spatial-temporal recommendation problems, supports easy integration of additional factors due to model independence, and enables separate optimization of accuracy and coverage with good scalability. Experiments on two real datasets demonstrate superior performance compared to baselines, proving its effectiveness for practical MCS applications. Future work will consider more influencing factors and explore faster training to capture preference changes promptly.

Note: Figure translations are in progress. See original paper for figures.