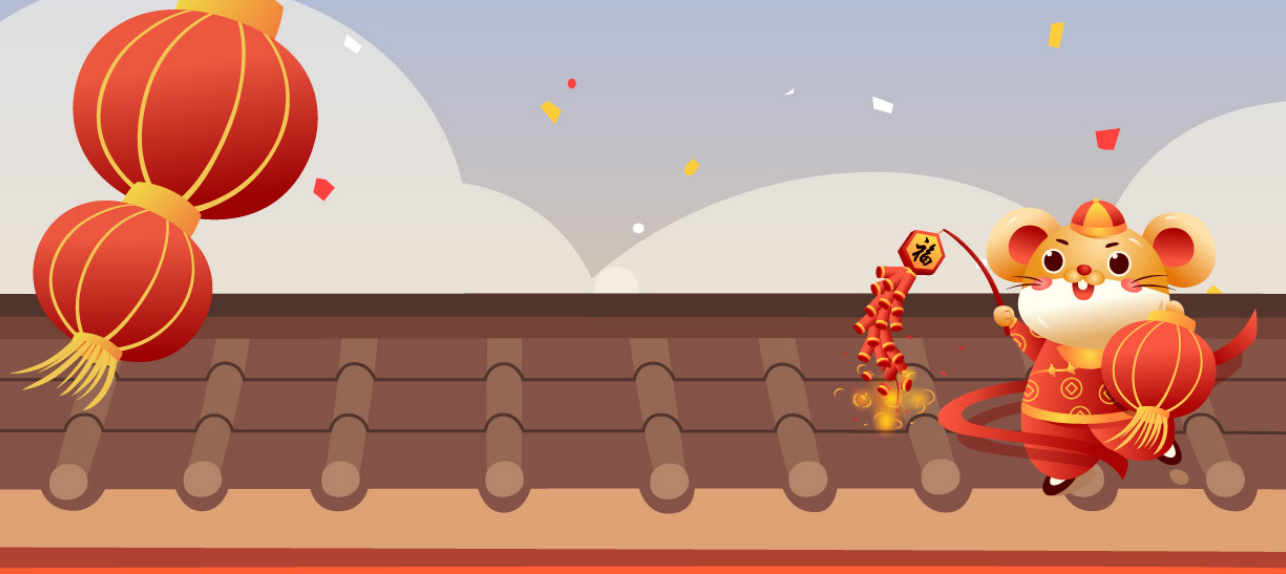


美团点评
2019 技术年货

CODE A BETTER LIFE

【学术论文篇】

美团点评

微信扫码关注技术团队公众号

tech.meituan.com
美团技术博客

新年
快乐

目录

论文精选

美团技术团队不仅在新技术落地上追求卓越，还积极探索前沿技术。2019 年，美团技术团队联合国内外多所高校和学者，在数个国际会议上发表重要论文，包括人工智能领域顶会 KDD、CVPR、IJCAI、WWW，GIS 顶会 SIGSPATIAL，机器人顶会 IROS，运筹学年会 INFORMS。

美团点评还在文字识别国际顶会 ICDAR 2019 发布了真实场景的中文门脸招牌图像数据集，并举办了中文门脸招牌文字识别比赛。

依托美团点评生活服务领域多样化真实场景及丰富数据，我们愿与学术界携手，协同创新，探索前沿技术方向，推动理论研究在产业实践中的落地。

The 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems

Effective Recycling Planning for Dockless Sharing Bikes

Cong Zhang
Beijing Uni. of Posts and Tele.
cong1126@bupt.edu.cn

Yanhua Li
Worcester Polytechnic Institute, USA
yli15@wpi.edu

Jie Bao
JD Finance
baojie@jd.com

Sijie Ruan
Xidian University
ruansijie@jd.com

Tianfu He
Harbin Institute of Technology
Tianfu.D.He@outlook.com

Hui Lu, Zhihong Tian
Guangzhou University
{luhui,tianzhihong}@gzhu.edu.cn

Cong Liu
The University of Texas at Dallas
cong@utdallas.edu

Chao Tian
Tencent
astortian@tencent.com

Jianfeng Lin, Xianen Li
Mobike
{linjianfeng,lixianen}@mobike.com

ABSTRACT

Bike-sharing systems become more and more popular in the urban transportation system, because of their convenience in recent years. However, due to the high daily usage and lack of effective maintenance, the number of bikes in good condition decreases significantly, and vast piles of broken bikes appear in many big cities. As a result, it is more difficult for regular users to get a working bike, which causes problems both economically and environmentally. Therefore, building an effective broken bike prediction and recycling model becomes a crucial task to promote cycling behavior. In this paper, we propose a predictive model to detect the broken bikes and recommend an optimal recycling program based on the large scale real-world sharing bike data. We incorporate the realistic constraints to formulate our problem and introduce a flexible objective function to tune the trade-off between the broken probability and recycled numbers of the bikes. Finally, we provide extensive experimental results and case studies to demonstrate the effectiveness of our approach.

CCS CONCEPTS

• **Applied computing** → *Transportation*; *Forecasting*; *Transportation*; • **Information systems** → *Spatial-temporal systems.*

KEYWORDS

bike-sharing systems, predictive model, optimal recycling program

ACM Reference Format:
Cong Zhang, Yanhua Li, Jie Bao, Sijie Ruan, Tianfu He, Hui Lu, Zhihong Tian, Cong Liu, Chao Tian, and Jianfeng Lin, Xianen Li. 2019. Effective Recycling Planning for Dockless Sharing Bikes. In *SIGSPATIAL '19: 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, November 5–8, 2019, Chicago, Illinois, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3347146.3359340

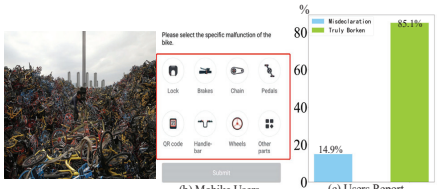
SIGSPATIAL '19, November 5–8, 2019, Chicago, Illinois, USA

ACM ISBN 978-1-4503-6909-1/19/11…$15.00
https://doi.org/10.1145/3347146.3359340

1 INTRODUCTION

Bike-sharing system is a popular transportation system in modern cities, as it not only provides an environment friendly choice for short-distance travelling, but also eases the traffic congestion. Currently, there are over 1,000 deployed bike-sharing systems world wide, and more than 300 systems are in the progress of deployment [29]. In recent years, station-less bike-sharing services, like Mobike[1], which allow users to pick up and drop off bikes at any locations they want, become more popular.

Due to the sharing nature of the bike-sharing systems, the sharing bikes have much higher broken possibilities compared with private bikes due to the high ridden frequency and open-air parking problem. For example, the bike sharing system in New York saw 3.6 daily rides per bike [2]. As a result, as shown in Figure 1(a), thousands of broken station-less sharing bikes are being kept in a bike graveyard.



(a) Bike Graveyards (b) Mobike Users Malfunction Report (c) Users Report Fault Bike Review

Figure 1: Issues with broken Sharing Bike.

Since the number of bikes put in the market is limited, without the proper maintenance, the number of bikes in good condition is continuously decreasing. The broken bikes not only cause economic losses to the companies but also lead to environmental pollution. Therefore, an effective bike recycling plan should be conducted. Currently, Mobike develops a broken bike report function in the app, so that the broken bikes can be discovered in a crowdsourcing way. As shown in Figure 1(b), users can report different types of bike problems in the mobile app, so that the company can arrange workers to collect and recycle them.

[1]https://en.wikipedia.org/wiki/Mobike
[2]https://bit.ly/2T6q5SE

However, there are *three* challenges to conduct such a broken bike recycling task:

Inaccurate and Inadequate Labels. Though the broken bike report function can help the company to quickly locate the broken bike, the report cannot be fully trusted. As shown in Figure 1(c), we manually exam the status of the reported broken bikes. Only 85.1% bikes are truly broken. Furthermore, not all of the users are willing to report the broken status of the bikes, as the broken report function is not a required step.

Arbitrary Spatial Distribution. Different from the station-based systems, the parking location of each individual station-less bike is totally arbitrary, which makes the recycling routes vary from day to day.

Limited Recycle Capacity. Given a set of bikes to be recycled, the worker can only collect the limited number of broken bikes within the working hour. Besides, the capacity of the collecting vehicle is limited, and the worker has to drive back to the recycling site as soon as the vehicle is full of broken bikes.

In this paper, we design a broken bike recycling route planning system for the worker. This system consists of two main modules: 1) broken bike inference, which infers the broken probability of each sharing bike using its inherent characteristics and the user trajectories associated with it; and 2) recycling route planning, which plans multiple closed recycling routes for the worker to conduct in each day.

The contributions of the paper are summarized as follows:

(1) We propose a novel broken sharing bike recycling problem, which takes the broken probability, working time constraint, and vehicle capacity into consideration.

(2) We build a broken bike inference model using inherent features and trajectory features extracted from the sharing bike so that the status of every single bike can be accurately inferred.

(3) We propose a scatter search-based heuristic algorithm for the broken sharing bike recycling problem.

(4) Experiments show the recycling efficiency of the broken bikes recommended by scatter search algorithm is 2.5 times that of the regional random search method and 1.5 times that of the Nearest neighbor routing search method. At the same time, the result of the algorithm is twice the efficiency of Mobike employees' broken bikes recycling.

The rest of the paper is organized as follows: Section 2 describes the problem and the system overview. Broken sharing bike inference model is discussed in Section 3. Section 4 gives the solution of broken sharing bike recycling routing problem. Experiments and case studies are given in Section 5. Related works are summarized in Section 6. Section 7 concludes the paper.

2 OVERVIEW

In this section, we define the broken prediction and recycling routing problem for Sharing Bike, and outline our solution framework.

2.1 Preliminaries

We define p_i as the inferred broken probability of sharing bike b_i. In the recycling task, we only consider bikes which are inferred as broken, i.e., $p_i > 0.5$. The bike with high broken probability is preferred to collect in the priority given limited working time. However, the bikes with high broken probability can distribute unevenly in the given region, which introduces large traveling time, and finally leads to less number of broken bike collected. As a result, we define a beneficial score $score_i$ below to characterize the worthiness of collecting a particular bike b_i. In the broken bike recycling mission, the dockless sharing bike can be at any location in the city, e.g., hiding in the residential area or close to the road network, where the parking location of collecting vehicles is usually along with the road network. As a result, the distance between them varies significantly, which we define v_w (walking speed) and rt (registration time) below to better characterize the individual bike collecting events.

Definition 1. ***(Beneficial score)*** *$score_i$ captures the overall benefit to recycle bike b_i, which characterizes the trade-off between the broken likelihood and the recycling cost of b_i.*

$$score_i = \alpha^{\frac{p_i}{\min p}} \qquad \alpha \geq 1 \tag{1}$$

where the parameter α represents the trade-off preference on the broken probability p_i vs recycling cost. $\min p$ is the minimum broken probability over all the bike in the region, which serves as a normalization term.

Each bike b_i has a broken probability p_i, i.e., the likelihood of being a broken bike. In practice, the trade off when choosing a bike is: If we seek for only bikes of high broken probability p_i, we may end up with a small number of bikes collected (less efficient); on the other hand, if we seek for a large number of collected bikes, many bikes collected may not be broken (false positive). The beneficial score defined in definition 1 captures such a trade-off by the parameter α. The reason for designing a score function using the exponential function is that the bike with higher broken probability will have a higher score ($\alpha > 1$). When α is close to 1, the efficiency is highly considered, leading to a large number of collected bikes; on the other hand, when $\alpha \gg 1$ is large, the broken probability p_i is highly considered, thus only bikes with high p_i will be collected. Especially, $\alpha = 1$ means that we do not care about the broken probability of the bike, and every broken bike has the same beneficial score. The α is a tunable parameter (chosen by the service operators), which provides them the flexibility between the efficiency (i.e., the number of collected bikes) and the likelihood of the collected bike being broken. From the operator's perspective, there are different objectives under various circumstances, for example, in regions hard to access, the efficiency should be highly considered (i.e., choosing α close to 1), while in areas with bikes densely populated, e.g., downtown, accurately collecting each broken bike is preferred, thus the likelihood of broken bikes needs to be considered more (i.e., choosing a large α). As a result, the beneficial score measures the practical "benefit" of collecting each bike.

Definition 2. ***(Sub-route)*** *Each closed route, which starts and ends at the collection site s, is considered as a sub-route.*

Definition 3. ***(Time Cost)*** *The time cost of sub-route R_j is composed of the vehicle travelling time between consecutive locations and the visiting time at each broken bike. Given a sub-route $R_j = s \rightarrow b_{r_1} \rightarrow \cdots \rightarrow b_{r_n} \rightarrow s$, the time cost T_j is calculated as follows:*

$$T_j = T_{travel}(R_j) + \sum_{i=1}^{n} T_{visit}(b_{r_i}). \tag{2}$$

Let us denote the shortest road network distance between broken bike b_i and b_j as $dist(b_i, b_j)$, and the vehicle driving speed as v_d, then the travelling time cost is calculated as follows:

$$T_{travel}(R_j) = \frac{dist(s, b_{r_1}) + \sum_{i=1}^{n-1} dist(b_{r_i}, b_{r_{i+1}}) + dist(b_{r_n}, s)}{v_d}. \tag{3}$$

The broken bike visiting time includes the walking time between the vehicle on the main road and the location of the broken bike, and the broken bike registration time rt. We denote the walking speed as v_w, and the perpendicular distance of the broken bike b_i to the nearest road segment as $shift_i$, then the visiting time cost can be represented as

$$T_{visit}(b_{r_i}) = \frac{2shift_{r_i}}{v_w} + rt. \tag{4}$$

Problem Definition. Given the road network RN, driving speed v_d, walking speed v_w, collection site s, broken bike registration time rt, working hour T, vehicle capacity M, and a broken sharing bike distribution graph $G = (V, E)$. The vertex set $V = \{b_1, b_2, \cdots, b_n\}$ represents all the broken bikes in the given service region of s, each of which is associated with a spatial location and a collection score $score_i$, and the edge set E denote the road network connectivity of broken bike pairs.

The objective of the broken bike recycling route planning problem aims to plan multiple traveling routes for the worker, so that the total score collected is maximized. The recycling route planning problem fulfills three constraints: (1) each broken bike is collected at most once; (2) the working time of the personnel is no more than T; and (3) the broken bikes collected in each sub-route are no more than the vehicle capacity M. If we use δ_{ij} to denote whether the broken bike b_i is collected during sub-route R_j, the problem can be formulated as follows:

$$\max_{\mathcal{R}} \quad \sum_{b_i \in V} \sum_{R_j \in \mathcal{R}} \delta_{ij} score_i \tag{5}$$

$$str. \quad \sum_{R_j \in \mathcal{R}} \delta_{ij} \leq 1, \quad \forall b_i \in V \tag{6}$$

$$\sum_{R_j \in \mathcal{R}} T_j \leq T \tag{7}$$

$$\sum_{b_i \in V} \delta_{ij} \leq M, \quad \forall R_j \in \mathcal{R} \tag{8}$$

Such a problem of finding k budget constrained connected components with a maximum beneficial score is NP-hard as proven in Lemma 1 below.

Lemma 1 (NP-difficulty). *When time and capacity constrained, collecting broken-sharing-bikes with a maximal beneficial score is NP-hard.*

Proof. The broken sharing bikes collection problem is a combination of broken sharing bike vertex selection and determining the shortest path between the selected vertices. As a consequence, We can reduce our problem of collecting broken-sharing-bikes with maximal beneficial score from the Knapsack Problem (KP) and the Travelling Salesperson Problem (TSP), when time and capacity constrained. We can view each broken sharing bike $b_i \in V$ as an item, with an item size (i.e., Collecting time cost), and an item profit (e.g., a beneficial score contribution). The set V of selected broken sharing bikes is viewed as a knapsack, with a fixed size T (i.e., total working time constraint). Furthermore, not all broken sharing bike $b_i \in V$ have to be visited in the problem. Determining the shortest path between the selected vertices $b_i \in V'$ will be helpful to visit as many vertices as possible in the available time. our goal is to maximize the total score collected. If a recycling worker with not enough time and capacity to collect all possible broken sharing bikes. He knows the number of beneficial scores to expect in each broken bike and wants to maximize the total beneficial score, while keeping the total travel time limited to T. Our problem boils down to an Orienteering Problem problem (OP), which is known to be NP-complete [41]. □

Given it is an NP-hard problem, we develop a heuristic-algorithm to tackle the issue.

2.2 System Overview

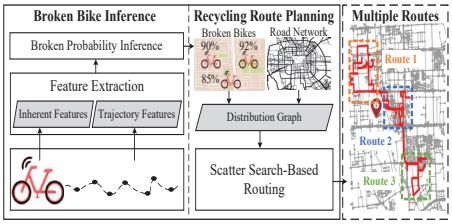


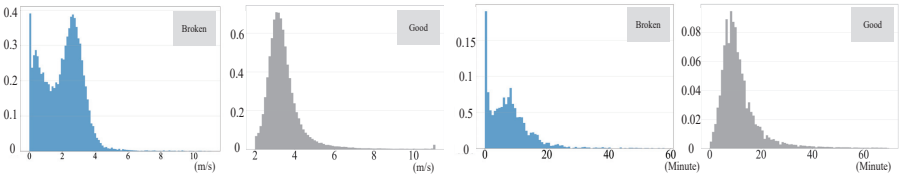
Figure 2: System Overview.

Figure 2 gives an overview of our system, which consists of two main components: (1) *Broken Bicycle Inference*, which calculates broken probability for each sharing bike, which takes the sharing bike's parameters, e.g., the bike inherent feature, and trajectory features, and outputs the bike broken probability and current status (detailed in Section 3) and (2) *Recycling Route Planning* component takes the results of the prediction model, the road network data and the recycling of historical data as input. It establishes the distribution graph of the broken bikes (detailed in Problem Definition) and recommends the optimal route for recycling the broken bike (detailed in Section 4).

3 BROKEN BIKE INFERENCE

Due to the fact that there is only a small proportion of sharing bikes reported as broken by the users, and not all of the reported bikes are truly broken, a broken bicycle inference model is required to detect the real broken bikes for the worker to collect. An inference model under the supervised-learning paradigm is used to assign a broken probability to each bicycle. In the later routing algorithm, the bike with high broken possibility is preferred to collect.

The training bike samples are selected as follows: 1) If a bike is reported as broken by Mobike user and the broken status is confirmed by the worker, we regard it as a broken bike sample; 2) If



(a) Difference of speed probability density distributions (b) Difference of duration probability density distributions

Figure 3: Mobike Trip Characteristics.

a bike is rode repeatedly in a time period (i.e., one month), and the user does not report the status of the bike as a broken, we regard it as a good bike sample.

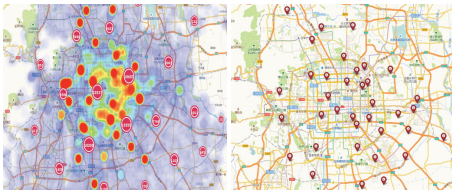
Feature Extraction. Whether a bike is broken can be inferred mainly from two aspects: *1) inherent features*, such as the life time of the bicycle, the number of ridden times, the total duration of cycling, and the number of maintenance; and *2) trajectory features*, which include the average travel speed and trip duration distributions. The selected trajectory features are derived from the analytics of Mobike trajectories. As shown in Figure 3(a), the probability of the average riding speed less than $1m/s$ for the broken bike is much higher than the good bike. This may be because some broken bikes are more cumbersome, the cycling speed will be slower. And from Figure 3(b), the trip duration of the broken bike is much shorter compared with the good one. This may be because the user finds that there is some problem with the bike after scanning the bicycle to ride, thereby terminating the cycling behavior. This phenomenon of user riding helps to determine the state of the sharing bike.

Broken Probability Inference. Since the sharing-bike status takes two values: good or broken (not good), we use a 0-1 valued binary variable y to denote the status outcome, where 1 stands for broken and 0 stands for good. We use p_i to denote the broken probability of the bike b_i. The probability depends on many factors, such as the trip duration and speed of a bike, etc. Such information can be encoded into a feature vector X_i, which is associated with the inherent features and the trajectory features of sharing bikes. Given extracted feature vector X_i, we can estimate the acceptance probability as: $p_i = p(y = broken|X_i)$. Since then, the broken inference task can be formulated as a typical binary classification problem, and the traditional classification model, such as Logistic Regression [11], can be employed.

4 RECYCLING ROUTE PLANNING

After the broken probability of each bike in the service region of a collection site is obtained, the distribution graph is constructed using bike locations with broken probabilities and the road network data. In this section, we describe the *scatter search-based routing algorithm* for the broken bike recycling problem using the constructed distribution graph.

In broken bike recycling problem, the instance size is surely beyond the solvability of standard solver, for example, as shown in Figure 4, there are typically hundreds of broken bikes in some regions, and 39 broken sharing bike collection site in Beijing. The collection



(a) Distribution of Broken Sharing Bikes (b) Distribution of Collection Site

Figure 4: Boken Mobike sharing Bike and collection site Distribution in Beijing

site of broken sharing bike need to occupy certain resources, so each collection site has its own service range. The departure and return locations of the workers are the same collection site in the area. If there is no limit to the capacity of the recycling vehicle and there are no restrictions on the working hours of the recycling workers. Our problem of recycling broken bikes with maximal beneficial score can be converted into a problem of recycling all broken sharing bikes and minimizing the overall recycling path, which can be converted into a tsp problem. However, in the case of working hours and the limited capacity of the recovered vehicle. The problem can be described as workers with not enough time and vehicle capacity to collect all possible broken bikes. He knows the beneficial score which is uniquely defined by the practical broken-bike collection problem (detailed in 2) of each broken bikes, and wants to maximize beneficial scores, while with the working hours and vehicle capacity limited.

Main Idea. Due to the capacity limitation M of the recycling vehicle, the worker can only collect the limited number of bikes during one sub-route. The main idea is that during each sub-route, we first try to collect at most M bikes with high broken probabilities (i.e., high beneficial scores), which are spatially close to each other, and then carefully plan the visiting order, so that the traveling time in each sub-route is minimized. We continuously find such sub-route until the working time is used up. The discovery of each sub-route is explained in following three stages: 1) *broken bike clustering*; 2) *sub-route selection*; 3) *status update*.

Stage 1: Broken Bike Clustering. In this stage, the bikes inferred to be broken are clustered using spatial clustering algorithm, e.g., kMeans [18], so that the broken bikes in each cluster are spatially close to each other. The number of clusters k is computed

according to both recycling vehicle capacity M and The total number of broken bikes in the area n. We initialize k as $k = round(n/M)$.

Stage 2: Sub-route Selection. In this stage, the algorithm finds the best sub-route in each cluster, and the best sub-route over all the clusters is selected. The goodness of a sub-route is defined as the beneficial score per time cost. The sub-route selection in each cluster is conducted in an iteratively way following the scatter search idea. We first select bikes with top M high scores as the initial bike set to recycle, and then design recycling route for it using TSP algorithm. Then we randomly replace a bike in the sub-route with a bike outside the sub-route but inside the cluster, to check whether there is any improvement. This process is repeated N times to obtain a stable sub-route in each cluster.

Stage 3: Status Update. In each iteration, the algorithm puts the best sub-route R_j into the final recycling route set $\mathcal{R}$, and updates the working time by subtracting the time spent recycling broken bikes in R_j and broken bike vertex set V by subtracting the recycled broken bikes in sub-route R_j. The algorithm terminates when working time T is used up, and then returns the recycling route set $\mathcal{R}$ as the recommended broken bikes recycling plan.

Algorithm Design. Algorithm 1 gives the pseudo-code of our scatter search-based heuristic algorithm. In each iteration of the *Scatter Search stage*(Line 2), the algorithm first partition the vertex set of broken bike nodes V into k clusters. The value of K is determined by the number of broken sharing bikes and the capacity of the recycling vehicle. Then, optimal broken sharing bikes collection scheme in the cluster is then selected separately in each independent cluster. When initializing the sub-route set in each cluster, two initialization strategies are employed depending on the value of a α. If the number of broken bikes in the candidate set S_i is greater than recycling vehicle capacity M, the initial recovery of the bicycle is selected using two methods. If tuning parameter α is equal to 1, the algorithm random select M broken bikes point in set S_i. otherwise, the algorithm select M broken bike in set S_i by the probability value of each broken bike as the candidate set C_i (Line 4-10). After selecting the initial result set C_i in cluster i, we use Function RecyRoute to solve the optimal recycling order of the broken bike in the result set and calculate the gain of recycling benefit score g_i. In the set S_i in which the number of each broken bicycle is larger than the recycling vehicle capacity, Take the broken bicycle not included in the set C_i which are randomly selected from the set S_i to replace random replace a broken bike in the set C_i. During the process, we keep track of the set C_i' and C_i, which has the maximum score gain in the iteration. If the number of broken bikes in the candidate set S_i is less than recycling vehicle capacity M, we just calculate the corresponding beneficial score gain. Select the best set C_i which has the maximum score gain from all clusters, and puts the best set R_j in recycling route set $\mathcal{R}$ base on Function RecyRoute. Then, R_i is removed from broken sharing bikes set V, the remaining working time is updated by subtracting the time cost $R_i.time$.At the same time, due to the reduction of the number of broken bikes, the number of clusters is also reduced (Line 11- 19).

Finally, when all the working time budget is used up, the algorithm terminates, and broken sharing-bikes recycling route set $\mathcal{R}$ is returned as the recommended broken bike recycling plan.

Algorithm 1 Scatter Search-based Routing Algorithm

Input: Broken sharing-bikes distribution graph $G = (V, E)$, working time T, parameter α, capacity M, initial number of clusters k and the maximum number of iterations N.
Output: Recycling route set $\mathcal{R}$.

1: **while** $T > 0$ **do**
//Stage 1: Broken Bike Clustering
2: $(\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_k) \leftarrow$ Kmeans(V, k)
//Stage 2: Sub-route Selection
3: **for** $i \leftarrow 1$ to k **do**
4: **if** $\mathcal{S}_i > M$ **then**
5: **if** $\alpha = 1$ **then**
6: Random select M points in $\mathcal{S}_i$ as C_i
7: **else**
8: Select the top M of broken probability in $\mathcal{S}_i$ as C_i
9: **else**
10: Select all point in $\mathcal{S}_i$ as C_i
11: $R_i, g_i \leftarrow$ RecyRoute(C_i)
12: **for** $l \leftarrow 1$ to N **do**
13: Randomly swap $b_m \in C_i$ by $b' \in \mathcal{S}_i - C_i$ as C_i'
14: $R_i', g_i' \leftarrow$ RecyRoute(C_i')
15: **if** $g_i' > g_i$ **then**
16: $C_i \leftarrow C_i'$; $R_i \leftarrow R_i'$; $g_i \leftarrow g_i'$
17: $j \leftarrow_i g_i$
//Stage 3: Status Update
18: $\mathcal{R} \leftarrow \mathcal{R} \cup \{R_j\}$; $T \leftarrow T - R_j.time$; $V \leftarrow V - R_j$
19: $k \leftarrow k - 1$
20: return $\mathcal{R}$

Function RecyRoute(C_i)
$R_i \leftarrow$ TSP(C_i); $g_i \leftarrow \frac{R_i.score}{R_i.time}$
return R_i, g_i

5 EXPERIMENTS

In this section, we conduct extensive experiments to evaluate the effectiveness of our system. We first describe the real dataset used in the paper. Then, we present comparison results with other baseline methods over different values of α and working time constraints. Finally, we present real-world case studies to evaluate our broken bike detection and recycling route planning algorithm.

5.1 Datasets

Road Networks. The road network data in Beijing and Guangzhou, China is collected from Open Street Map [3].

Mobike Order Data. Each Mobike order contains a bike ID, a user ID. The dataset used in the paper includes the entire Mobike orders in the City of Beijing and Guangzhou from 01/08/2018 to 12/31/2018.

Mobike Recycling Data. Each Mobike recycling record contains a bike ID, a worker ID, the start time and the end time to recycle the bike. The dataset is collected in the City of Beijing, with the time span of 01/06/2017 - 12/31/2018.

Mobike Trajectories. Each Mobike trajectory contains a bike ID, a user ID, the time interval of the trajectory, the start/end locations, and a sequence of intermediate GPS points. The dataset includes the

[3] https://www.openstreetmap.org/

entire Mobike trajectory data in the City of Beijing and Guangzhou from 01/08/2018 to 12/31/2018.

5.2 Data Pre-Processing

Data Pre-processing takes the road network, the Mobike order data, Mobike recycling data, and the Mobike trajectories as input, and performs the following three tasks to prepare the data for further processing:

Data Cleaning. *Data Cleaning* cleans the raw order data, trajectories, and recycling data from Mobike. Essentially as a type of crowdsensing data, Mobike trajectories are generated by the GPS modules from mobile phones. As a result, a noticeable portion of trajectories has different data errors, which significantly affect the accuracy of the broken bike inference model. This step cleans the raw trajectories from Mobike users by filtering the noisy GPS points with a heuristic-based outlier detection method [43].

Map-Matching. In this module, we map the GPS points onto the corresponding segments in road networks, which is crucial for the broken sharing bike collection. The Mobike sharing bike can be at any location in the city, e.g., hiding in the residential area or close to the road network, where the parking location of collecting vehicles is usually along with the road network. As a result, we should employ v_w (walking speed) to better characterize the individual bike collecting events. This step evaluates the distance of each broken sharing bikes to the nearest corresponding segments in road networks with a global map matching method [28].

Map Griding. For the ease of assessing the regional rt (registration time), we adopt the griding based method, which simply partitions the map into equal side-length grids [23, 24]. our approach divides the urban area into equal-size grids with a pre-defined side-lengths in 100 meters.

5.3 Effectiveness Evaluation

In this subsection, we study the effectiveness of both broken bike prediction and recycling. Unless mentioned otherwise, the default parameters used in the experiments are: recycling vehicle capacity $M = 20$, the average speed of the worker's walking is $v_w = 1m/s$, and the average speed of the worker's driving is $v_d = 25km/h$.

5.3.1 Broken Bike Prediction.

In the broken bike prediction model, we tried two popular models: logistic regression (LR) [11] and random forest (RF) [2] algorithms. We train the models for different cities and evaluate both methods in terms of Accuracy (ACC) and Area under the Curve of ROC (AUC). Experimental results for Beijing and Guangzhou are shown in Table 1, where we observe that 1) LR outperforms RF slightly and 2) both models get good results, which validates the effectiveness of our feature extraction scheme.

5.3.2 Performance of Different TSP Methods in Recycling Route Planning.

We study the effect of different TSP methods in our recycling route planning. The test data select from Haidian District, Beijing, which the inference model give 537 broken bikes in this area as shown in Figure 7. In this work, we tried five popular models: Simulated Annealing Algorithms (SA) [13], Genetic Algorithms (GA) [31], Ant Colony Optimizations Algorithms (ACO)[9], Lin-Kernighan(LK)[14] and Self-organizing Feature Maps (SOFM)[3].

Table 1: Results of LR and RF

Beijing				
Model	ACC	AUC	Recall	F-score
LR	0.9768	0.9965	0.9763	0.97796
RF	0.9750	0.9934	0.9746	0.97608
Guangzhou				
Model	ACC	AUC	Recall	F-score
LR	0.9757	0.9948	0.9759	0.9756
RF	0.9746	0.9933	0.9745	0.9750

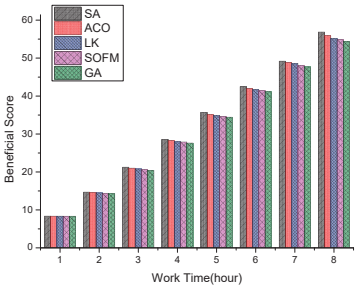


Figure 5: The Evaluation of Different TSP Methods.

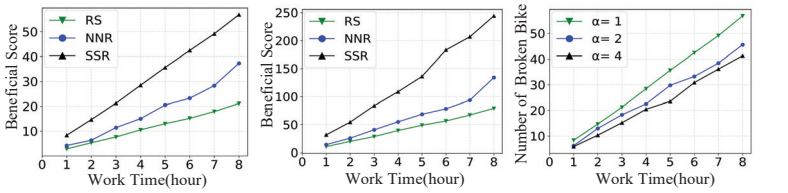
We evaluate five methods in terms of the total beneficial scores in our recycling broken sharing bike problem. Experimental results for Beijing are shown in Figure 5. Our experiments show that for our test data, these TSP algorithms do not make a significant difference as well. SA is slightly more accurate. Therefore, we choose SA as the TSP method in our recycling route planning model.

5.3.3 Recycling Route Planning.

We study the effect of different parameter settings of α and working time, and we compare our method, i.e. Scatter Search-based Routing (SSR), with two other baselines.

• **Baseline 1: Random selection (RS).** If there is no inference model in the collection problem, we assume that workers collect broken bikes according to user reports which occur randomly. We directly take the next car after each collection of a broken bike for collection. When the number of broken bikes collected reaches the vehicle capacity, return to the broken bike station in the area and repeat the random collection process for the next round. The collection process terminates when the total collection time exceeds the working time.

• **Baseline 2: Nearest neighbor routing (NNR).** The location where the broken bike is relatively densely distributed is selected as the starting area for collecting the first broken bike. In NNR, the recycling vehicle starts at the recycling parking spot, repeatedly visits the nearest broken bikes node until the capacity of the vehicle and the working hours of the workers exceed the constraint and returns back to the parking spot.



(a) Results with α =1 (b) Results with α =2 (c) Result of Recycling of Different α

Figure 6: Effectiveness Evaluation.

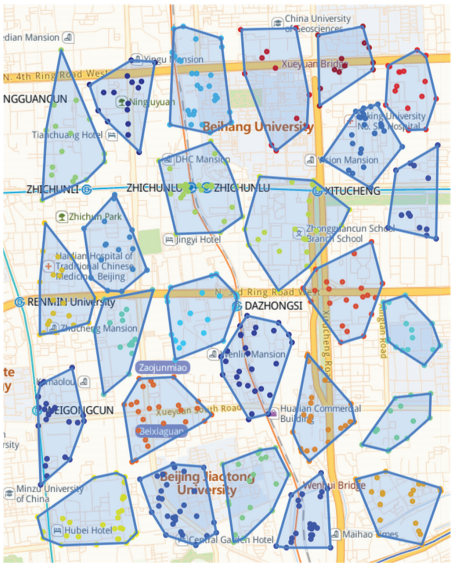


Figure 7: Broken sharing bike distribution cluster in Haidian District.

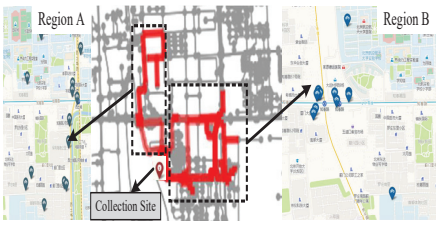


Figure 8: A Real Case Study in Haidian District, Beijing.

Effects on Total Working Time Budget. Figure 6 illustrates the total beneficial scores with different total working time budgets for a worker with Mobike, from 1 Hour to 8 Hour. The experimental results of finding a broken bike based on a random walk of RS is the average value of the income score after the algorithm solves the problem 1000 times independently. From the figure, we make the following observations: 1) the scatter search-based heuristic SSR method performs better than other baseline models. 2) When working hours are between 5 hours and 6 hours, the NNR method will have a useful period of slow growth. This is because, during this time, the NNR method took a long time in a broken bicycle. It is interesting that, because of the slight damage to the bike, the user is not quite sensitive to it and the bike moves within a certain range. This has resulted in a higher recovery cost for workers. The SSR method can adjust the recovery of the beneficial score by controlling the parameter α, which can solve the phenomenon and make the overall recovery more effective as shown in Figure 6(b). Figure 6(c) provides the results with different α settings, It is interesting that, when α is large, the number of recycling broken sharing bikes will be reduced to some extent. Moreover, with a higher α, the number of recycling broken bikes is smaller, but the degree of reduction will gradually decrease. The reason behind these phenomena is that a bicycle with a higher probability of failure prediction has a higher score. Where the value of α is larger, and it is more preferable to collect a bicycle which has higher broken probability when collecting broken sharing bikes. However, the distance between bikes also affects the time cost of recycling, so when the value of a is larger, the number of broken bikes collected will become smaller.

5.4 Case Studies

To better understand the effectiveness of our bike prediction and recycling model, we conduct a field case study. We choose to visit the area near Zhichun Road, Dazhongsi, and Beitucheng subway station in Haidian District, Beijing.

Figure 8 gives the path that Mobike operators use to recycle broken bikes in this area. The workers recovered a total of 32 broken bikes in the vicinity in 8 hours, and mainly concentrated near the temporary parking spots. The traditional recycling methods of worker are similar to the NNS method. The worker first finds the area where the broken bikes are densely distributed near the temporary parking point through the location reported by bikes. Then,

a broken bike in the dense area is selected and recycled according to the scheme of the shortest travel distance of the map navigation. When the target bicycle is found, another broken bike which is closest to the current location is selected for recycling. However, this will make recyclers tend to pay more attention to broken bikes that are closer to temporary parking spots. The distribution of broken bikes in the area is not fully considered, resulting in low efficiency and high cost of recycling. Figure 9 shows the results of the broken recovery path recommended by the SSR method. Mobike's worker recycles the broken bikes in a given area three times and collect 59 broken bikes in an 8 hour working time limit. It is interesting that, the recommended recovery path of the algorithm is not only the broken bikes concentrated near the temporary parking point, but also the broken bikes far from the temporary parking point are also included in the recommended collection of recycling. This is because the algorithm parameters take into account of the overall distribution of the bikes and the optimal recovery sequence in the actual recycling process, and the parameters are tuned according to the efficiency gain of each bike recovery. This makes the recycling of bikes more efficient. At the same time, the two broken bikes at the bottom left of Figure 9 are the broken bikes that are seriously broken in the recovery. One is the chain is broken, and the other is the seat is lost. The confidence of the two cars in the model is 0.99988294 and 0.99961954 respectively. These two bikes belong to the broken bike that is preferentially collected when the value of α is greater than one. This shows that the model is sensitive to bike with a relatively high degree of failure.

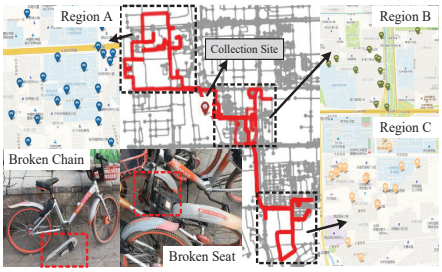


Figure 9: A Real Case Study Base on Scatter Search Model Result in Haidian District, Beijing.

6 RELATED WORK

The research of fault sharing bikes recycling can be summarized in two main areas: 1) Urban Crowd Sourcing, and 2) Route Planning.

Urban Crowd Sourcing. Essentially, we take advantage of the massive Mobike users in a city to perform the fault bike detection task. Similar problems are addressed with the crowdsourcing techniques[19, 26]. For example, The literature [42] quantifies the fragility of cities through detecting the delay in commuting activities using GPS data collected from smartphones. The literatures [34, 36] infer noise levels for locations by smartphone users. The literature [27] proposes a bike sharing network optimization approach by extracting fine-grained discriminative features from human mobility data, point of interests (POI), as well as station network structures. The literatures [7, 15] identify potholes or classify road quality from vehicle's accelerometer data. Differing from the above works, we focus on the problem of broken sharing bikes detection and collecting path planning.

Route Planning. The fault sharing bike recycling problem is related to the multiple Traveling Salesman Problem (mTSP) [1, 38] and orienteering Problem (OP) [4, 39, 41]. mTSP and OP can be considered as a relaxation of our problem, with the capacity or working time restrictions removed. The solutions for these two problems are primarily in two fold: 1) optimal algorithms and 2) heuristic algorithms. In literatures [21, 35], the authors use branch-and-bound to solve instances with less than 20 and 150 vertices, respectively. The authors in [22] use a cutting plane method to obtain better upper bounds. In literatures [10, 12], the authors propose branch-and-cut algorithms. However, the branch-and-cut procedure with instances up to 500 vertices cannot be performed. GAs are relatively stochastic search algorithms based on evolutionary biology and computer science principles [16]. Using GAs to the mTSP problem have several representations, like one chromosome technique [33], the two chromosome technique [32] and the latest two-part chromosome technique. The authors in [25] propose an ant colony optimization approach and a tabu search algorithm. In literature [37], the authors develop a Pareto ant colony optimization algorithm and a multi-objective variable neighborhood search algorithm. In [40], the authors propose a Variable Neighbourhood Search (VNS) algorithm and embed an exact algorithm to deal with a path feasibility subproblem. In [20], the authors present two polynomial size formulations for OP. The authors in [30] discuss several vehicle routing algorithms, and present a heuristic method which searches over a solution space formed by the large number of feasible solutions to an mTSP. The authors in [17] study the adaptive stochastic knapsack problem with deterministic size and stochastic rewards. Their problem objective is to find a sequential inserting policy to maximize the probability of the reward exceeding a threshold value without violating the capacity constraint. In [5], the authors study the adaptive stochastic knapsack problem with items of deterministic reward and stochastic size. Their goal is to maximize expected value while fitting all the items in the knapsack. The authors demonstrate the benefit of an adaptive policy and provide an approximation approach. In [6], the authors study an orienteering problem with stochastic travel times and present adaptive path planning methods to take advantage of dynamically updating data; combine the orienteering problem and optimal path finding into a single model. The authors in [8] discuss the vehicle routing problem with hard time windows and stochastic service times (VRPTW-ST). They adopt the dynamic programming algorithm to account for the probabilistic resource consumption by extending the label dimension and by providing new dominance rules. In this paper two recourse strategies are proposed and the resulting problems are solved by branch-price-and-cut algorithms. However, all of these works cannot be directly used for broken sharing bikes recycling, because these works simply test on benchmark instances and fail to consider the realistic constraints and road network distance.

7 CONCLUSION

In this paper, we introduce a novel approach to detect broken sharing bikes and recommend the appropriate bicycle recycling path to the worker based on the real sharing bikes data collected from Mobike (a major station-less bike sharing system). Our system can address the problem of recycling efficiency of broken sharing bikes in a more realistic fashion, considering the constraints and requirements from sharing bike worker's perspective: 1) working time limitations, 2) vehicle capacity constraints, and 3) broken sharing bike recovery benefit. We also propose a flexible beneficial score function to adjust preferences between the number of bikes recovered and the predicted probability of damage to bikes. The formulated problem is proven to be NP-hard, thus we propose a *scatter search-based heuristic* algorithm. We perform extensive experiments on a large scale Mobike data and demonstrate the effectiveness of our proposed broken sharing bike predict model and bike recycling routing model, where our model can predict the broken sharing bikes with above 97% accuracy and recommends that the number of real broken bikes recovered by the recycling path of the broken bikes is two to three times that of the Mobike traditionally recycling broken bikes.

REFERENCES

[1] Tolga Bektas. 2006. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega* 34, 3 (2006), 209–219.

[2] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[3] Łukasz Brocki and Danijel Koržinek. 2007. Kohonen self-organizing map for the traveling salesperson problem. In *Recent Advances in Mechatronics*. Springer, 116–119.

[4] I-Ming Chao, Bruce L Golden, and Edward A Wasil. 1996. The team orienteering problem. *European journal of operational research* 88, 3 (1996), 464–474.

[5] Brian C Dean, Michel X Goemans, and Jan Vondrdk. 2004. Approximating the stochastic knapsack problem: The benefit of adaptivity. In *45th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 208–217.

[6] Irina Dolinskaya, Zhenyu Edwin Shi, and Karen Smilowitz. 2018. Adaptive orienteering problem with stochastic travel times. *Transportation Research Part E: Logistics and Transportation Review* 109 (2018), 1–19.

[7] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, and Hari Balakrishnan. 2008. The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*. ACM, 29–39.

[8] Fausto Errico, Guy Desaulniers, Michel Gendreau, Walter Rei, and L-M Rousseau. 2018. The vehicle routing problem with hard time windows and stochastic service times. *EURO Journal on Transportation and Logistics* 7, 3 (2018), 223–251.

[9] Jose B Escario, Juan F Jimenez, and Jose M Giron-Sierra. 2015. Ant colony extended: experiments on the travelling salesman problem. *Expert Systems with Applications* 42, 1 (2015), 390–410.

[10] Matteo Fischetti, Juan Jose Salazar Gonzalez, and Paolo Toth. 1998. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing* 10, 2 (1998), 133–148.

[11] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York.

[12] Michel Gendreau, Gilbert Laporte, and Frederic Semet. 1998. A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks: An International Journal* 32, 4 (1998), 263–273.

[13] Xiutang Geng, Zhihua Chen, Wei Yang, Deqian Shi, and Kai Zhao. 2011. Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search. *Applied Soft Computing* 11, 4 (2011), 3680–3689.

[14] Keld Helsgaun. 2009. General k-opt submoves for the Lin–Kernighan TSP heuristic. *Mathematical Programming Computation* 1, 2-3 (2009), 119–163.

[15] Marius Hoffmann, Michael Mock, and Michael May. 2013. Road-quality classification and bump detection with bicycle-mounted smartphones. In *Proceedings of the 3rd International Conference on Ubiquitous Data Mining-Volume 1088*. CEUR-WS. org, 39–43.

[16] John Holland. 1975. Adaptation in natural and artificial systems: an introductory analysis with application to biology. *Control and artificial intelligence* (1975).

[17] Taylan İlhan, Seyed MR Iravani, and Mark S Daskin. 2011. The adaptive knapsack problem with stochastic rewards. *Operations research* 59, 1 (2011), 242–248.

[18] Anil K Jain. 2010. Data clustering: 50 years beyond K-means. *Pattern recognition letters* 31, 8 (2010), 651–666.

[19] Shenggong Ji, Yu Zheng, and Tianrui Li. 2016. Urban Sensing Based on Human Mobility. UbiComp 2016. https://www.microsoft.com/en-us/research/publication/urban-sensing-based-human-mobility/

[20] Imdat Kara, Papatya Sevgin Bicakci, and Tusan Derya. 2016. New formulations for the orienteering problem. *Procedia Economics and Finance* 39 (2016), 849–854.

[21] Gilbert Laporte and Silvano Martello. 1990. The selective travelling salesman problem. *Discrete applied mathematics* 26, 2-3 (1990), 193–207.

[22] Adrienne C Leifer and Moshe B Rosenwein. 1994. Strong linear programming relaxations for the orienteering problem. *European Journal of Operational Research* 73, 3 (1994), 517–523.

[23] Yanhua Li, Jun Luo, Chi-Yin Chow, Kam-Lam Chan, Ye Ding, and Fan Zhang. 2015. Growing the charging station network for electric vehicles with trajectory data analytics. In *2015 IEEE 31st International Conference on Data Engineering*. IEEE, 1376–1387.

[24] Yanhua Li, Moritz Steiner, Jie Bao, Limin Wang, and Ting Zhu. 2014. Region sampling and estimation of geosocial data with dynamic range calibration. In *2014 IEEE 30th International Conference on Data Engineering*. IEEE, 1096–1107.

[25] Yun-Chia Liang, Sadan Kulturel-Konak, and Alice E Smith. 2002. Meta heuristics for the orienteering problem. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, Vol. 1. IEEE, 384–389.

[26] Dongyu Liu, Di Weng, Yuhong Li, Jie Bao, Yu Zheng, Huamin Qu, and Yingcai Wu. 2016. Smartadp: Visual analytics of large-scale taxi trajectories for selecting billboard locations. *IEEE transactions on visualization and computer graphics* 23, 1 (2016), 1–10.

[27] J. Liu, Q. Li, M. Qu, W. Chen, J. Yang, H. Xiong, H. Zhong, and Y. Fu. 2015. Station Site Optimization in Bike Sharing Systems. In *2015 IEEE International Conference on Data Mining*. 883–888. https://doi.org/10.1109/ICDM.2015.99

[28] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. 2009. Map-matching for low-sampling-rate GPS trajectories. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. ACM, 352–361.

[29] Russell Meddin and Paul DeMaio. 2015. The bike-sharing world map. (2015). URLhttp://www.bikesharingworld.com

[30] RH Mole, DG Johnson, and K Wells. 1983. Combinatorial analysis for route first-cluster second vehicle routing. *Omega* 11, 5 (1983), 507–512.

[31] Yuichi Nagata and Shigenobu Kobayashi. 2013. A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem. *INFORMS Journal on Computing* 25, 2 (2013), 346–363.

[32] Yang-Byung Park. 2001. A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines. *International Journal of Production Economics* 73, 2 (2001), 175–188.

[33] Jean-Yves Potvin, Guy Lapalme, and Jean-Marc Rousseau. 1989. A generalized k-opt exchange procedure for the MTSP. *INFOR: Information Systems and Operational Research* 27, 4 (1989), 474–481.

[34] Zhaokun Qin and Yanmin Zhu. 2016. NoiseSense: A crowd sensing system for urban noise mapping service. In *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 80–87.

[35] R Ramesh, Yong-Seok Yoon, and Mark H Karwan. 1992. An optimal algorithm for the orienteering tour problem. *ORSA Journal on Computing* 4, 2 (1992), 155–165.

[36] Rajib Kumar Rana, Chun Tung Chou, Salil S Kanhere, Nirupama Bulusu, and Wen Hu. 2010. Ear-phone: an end-to-end participatory urban noise mapping system. In *Proceedings of the 9th ACM/IEEE international conference on information processing in sensor networks*. ACM, 105–116.

[37] Michael Schilde, Karl F Doerner, Richard F Hartl, and Guenter Kiechle. 2009. Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence* 3, 3 (2009), 179–201.

[38] Joseph A Svestka and Vaughn E Huckfeldt. 1973. Computational experience with an m-salesman traveling salesman algorithm. *Management Science* 19, 7 (1973), 790–799.

[39] Tommy Thomadsen and Thomas K Stidsen. 2003. The quadratic selective travelling salesman problem. (2003).

[40] Fabien Tricoire, Martin Romauch, Karl F Doerner, and Richard F Hartl. 2010. Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research* 37, 2 (2010), 351–367.

[41] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. 2011. The orienteering problem: A survey. *European Journal of Operational Research* 209, 1 (2011), 1–10.

[42] Takahiro Yabe, Kota Tsubouchi, and Yoshihide Sekimoto. 2017. CityFlowFragility: Measuring the Fragility of People Flow in Cities to Disasters using GPS Data Collected from Smartphones. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3 (2017), 117.

[43] Yu Zheng. 2015. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)* 6, 3 (2015), 29.

2019 IEEE/RSJ International Conference on Intelligent Robots and Systems

StarNet: Pedestrian Trajectory Prediction using Deep Neural Network in Star Topology

Yanliang Zhu, Deheng Qian, Dongchun Ren, Huaxia Xia

***Abstract*— Pedestrian trajectory prediction is crucial for many important applications. This problem is a great challenge because of complicated interactions among pedestrians. Previous methods model only the pairwise interactions between pedestrians, which not only oversimplifies the interactions among pedestrians but also is computationally inefficient. In this paper, we propose a novel model StarNet to deal with these issues. StarNet has a star topology which includes a unique hub network and multiple host networks. The hub network takes observed trajectories of all pedestrians to produce a comprehensive description of the interpersonal interactions. Then the host networks, each of which corresponds to one pedestrian, consult the description and predict future trajectories. The star topology gives StarNet two advantages over conventional models. First, StarNet is able to consider the collective influence among all pedestrians in the hub network, making more accurate predictions. Second, StarNet is computationally efficient since the number of host network is linear to the number of pedestrians. Experiments on multiple public datasets demonstrate that StarNet outperforms multiple state-of-the-arts by a large margin in terms of both accuracy and efficiency.**

I. INTRODUCTION

Pedestrian trajectory prediction is an important task in autonomous driving [1], [2], [3] and mobile robot applications [4], [5], [6]. This task allows an intelligent agent, e.g., a self-driving car or a mobile robot, to foresee the future positions of pedestrians. Depending on such predictions, the agent can move in a safe and smooth route.

However, pedestrian trajectory prediction is a great challenge due to the intrinsic uncertainty of pedestrians' future positions. In a crowded scene, each pedestrian dynamically changes his/her walking speed and direction, partly attributed to his/her interactions with surrounding pedestrians.

To make an accurate prediction, existing algorithms focus on making full use of the interactions between pedestrians. Early works model the interactions [7], [8], [9], [10] by handcrafted features. Social Force [7] models several force terms to predict human behaviors. The approach in [8] constructs an energy grid map to describe the interactions in crowded scenes. However, their performances are limited by the quality of manually designed features. Recently, data-driven methods have demonstrated their powerful performance [11], [12], [13], [14]. For instance, Social LSTM [11] considers interactions among pedestrians close to each other. Social GAN [13] models interactions among all pedestrians. Social Attention [14] captures spatio-temporal interactions.

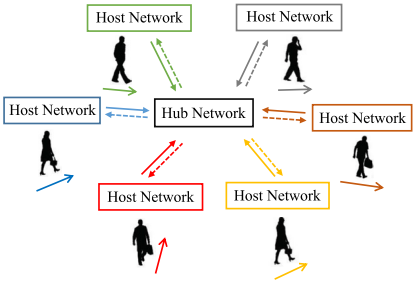


Fig. 1: The structure of StarNet. StarNet mainly consists a centralized hub network and several host networks. The hub network collects movement information and generates a feature which describes joint interactions among pedestrians. Each host network, corresponding to a certain pedestrian, queries the hub network and predicts the pedestrian's trajectory.

Previous methods have achieved great success in trajectory prediction. However, all these methods assume that the complicated interactions among pedestrians can be decomposed into pairwise interactions. This assumption neglects the collective influence among pedestrians in the real world. Thus previous methods tend to fail in complicated scenes. In the meanwhile, the number of pairwise interactions increases quadratically as the number of pedestrians increases. Hence, existing methods are computationally inefficient.

In this paper, we propose a new deep neural network, StarNet, to model complicated interactions among all pedestrians together. As shown in Figure 1, StarNet has a star topology, and hence the name. The central part of StarNet is the hub network, which produces a representation $\mathbf{r}$ of the interactions among pedestrians. To be specific, the hub network takes the observed trajectories of all pedestrians and produces a comprehensive spatio-temporal representation $\mathbf{r}$ of all interactions in the crowd. Then, $\mathbf{r}$ is sent to each host network. Each host network predicts one pedestrian's trajectory. Specifically, depending on $\mathbf{r}$, each host network exploits an efficient method to calculate the pedestrian's

*This work was supported by the Meituan-Dianping Group.

Yanliang Zhu, Deheng Qian, Dongchun Ren and Huaxia Xia are with the Meituan-Dianping Group, Beijing, China. zhuyanliang@meituan.com

interactions with others. Then, the host network predicts one pedestrian's trajectory depending on his/her interactions with others, as well as his/her observed trajectory.

StarNet has two advantages over previous methods. First, the representation $\mathbf{r}$ is able to describe not only pairwise interactions but also collective ones. Such a comprehensive representation enables StarNet to make accurate predictions. Second, the interactions between one pedestrian and others are efficiently computed. When predicting all pedestrians' trajectories, the computational time increases linearly, rather than quadratically, as the number of pedestrians increases. Consequently, StarNet outperforms multiple state-of-the-arts in terms of both accuracy and computational efficiency.

Our contributions are two-folded. First, we propose to describe collective interactions among pedestrians, which results in more accurate predictions. Second, we devise an interesting topology of the network to take advantage of the representation $\mathbf{r}$, leading to computational efficiency.

The rest of this paper is organized as follows: Section II briefly reviews related work on pedestrian trajectory prediction. Section III formalizes the problem and elaborates our method. Section IV compares StarNet with state-of-the-arts on multiple public datasets. Section V draws our conclusion.

II. RELATED WORK

Our work mainly focuses on human path prediction. In this section, we give a brief review of recent researches on this domain.

Pedestrian path prediction is a great challenge due to the uncertainty of future movements [7], [8], [10], [11], [13], [14], [15]. Conventional methods tackle this problem with manually crafted features. Social Force [7] extracts force terms, including self-properties and attractive effects, to model human behaviors. Another approach [8] constructs an energy map to indicate the traffic capacity of each area in the scene, and uses a fast matching algorithm to generate a walking path. Mixture model of Dynamic pedestrian-Agents (MDA) [10] learns the behavioral patterns by modeling dynamic interactions and pedestrian beliefs. However, all these methods can hardly capture complicated interactions in crowded scenes, due to the limitation of hand-crafted features.

Data-driven methods remove the requirement of hand-crafted features, and greatly improve the ability to predict pedestrian trajectories. Some attempts [11], [13], [14], [26], [27] receive pedestrian positions and predict determined trajectories. Social LSTM [11] devises social pooling to deal with interpersonal interactions. Social LSTM divides pedestrian's surrounding area into grids, and computes pairwise interactions between pedestrians in a grid. Compared with Social LSTM, other approaches [13], [15] eliminate the limitation on a fixed area. Social GAN [13] combines Generative Adversarial Networks (GANs) [16] with LSTM-based encoder-decoder architecture, and sample plausible trajectories from a distribution. Social Attention [14] estimates multiple Gaussian distributions of future positions, then generates candidate trajectories through Mixture Density Network (MDN) [17].

However, existing methods compute pairwise features, and thus oversimplified the interactions in the real word environment. Meanwhile, they suffer from a huge computational burden in crowded scenes. In contrast, our proposed StarNet with novel architecture is capable of capturing joint interactions over all pedestrians, which is more accurate and efficient.

III. APPROACH

In this section, we first describe the formulation of the pedestrian prediction problem. Then we provide the details of our proposed method.

A. Problem Formulation

We assume the number of pedestrians is N. The number of observed time steps is T_{obs}. And the number of time steps to be predicted is T_{pred}. For the i-th pedestrian, his/her observed trajectory is denoted as $O_i = \{\mathbf{p}_i^t \mid t = 1, 2, \cdots, T_{obs}\}$, where $\mathbf{p}_i^t$ represents his/her coordinates at time step t. Similarly, the future trajectory of ground truth is denoted as $F_i = \{\mathbf{p}_i^t \mid t = T_{obs}+1, T_{obs}+2, \cdots, T_{obs}+T_{pred}\}$.

Given such notations, our goal is to build a fast and accurate model to predict the future trajectories $\{F_i\}_{i=1}^N$ of all pedestrians, based on their observed trajectories $\{O_i\}_{i=1}^N$. In other words, we try to find a function mapping from $\{O_i\}_{i=1}^N$ to $\{F_i\}_{i=1}^N$. We employ a deep neural network, which is called StarNet, to embody this function. Specifically, StarNet consists of two novel parts, i.e., a hub network and N host networks. The hub network computes a representation $\mathbf{r}$ of the crowd. Then, each host network predicts the future trajectory of one pedestrian depending on the pedestrian's observed trajectory and $\mathbf{r}$. We first describe the hub network and then present host networks.

B. The hub network

The hub network takes all of the observed trajectories simultaneously and produces a comprehensive representation $\mathbf{r}$ of the crowd of pedestrians. The representation $\mathbf{r}$ includes both spatial and temporal information of the crowd, which is the key to describe the interactions among pedestrians.

Note that our algorithm should be invariant against isometric transformation (translation and rotation) of the pedestrians' coordinates. The invariance against rotation is achieved by randomly rotate our training data during the training process. While the invariance against translation is guaranteed by calculating a translation invariant representation $\mathbf{r}$.

As shown in Figure 2, the hub network produces $\mathbf{r}$ by two steps. First, the hub network produces a spatial representation of the crowd for each time step. The spatial representation is invariant against the translation of the coordinates. Then, the spatial representation is fed into a LSTM to produce the spatio-temporal representation $\mathbf{r}$.

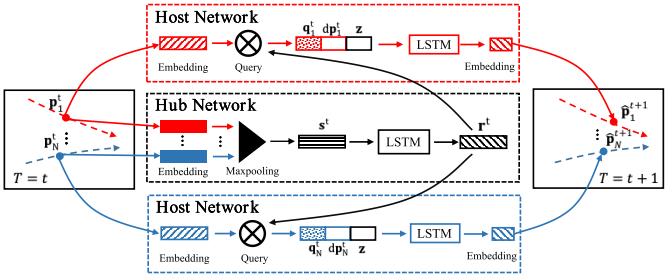


Fig. 2: The process of predicting the coordinates. At time step t, StarNet takes the newly observed (or predicted) coordinates $\{\mathbf{p}_i^t\}_{i=1}^N$ (or $\{\widehat{\mathbf{p}}_i^t\}_{i=1}^N$) and outputs the predicted coordinates $\{\widehat{\mathbf{p}}_i^{t+1}\}_{i=1}^N$.

1) Spatial representation: In the first step, in order to make the representation invariant against translation, the hub network preprocesses the coordinates of pedestrians by subtracting the central coordinates of all pedestrians at time step T_{obs} from every coordinate.

$$\mathbf{p}_i^t \leftarrow \mathbf{p}_i^t - \frac{1}{N}\sum_{n=1}^{N}\mathbf{p}_n^{T_{obs}}. \tag{1}$$

Thus, the centralized coordinates are invariant against translation. Such coordinates of each pedestrian are mapped into a new space using an embedding function $\phi(\cdot)$ with parameters W_1,

$$\mathbf{e}_i^t = \begin{cases} \phi\left(\mathbf{p}_i^t; W_1\right), & if\ t \in [1, T_{obs}], \\ \phi\left(\widehat{\mathbf{p}}_i^t; W_1\right), & if\ t \in \left[T_{obs}+1, T_{obs}+T_{pred}\right], \end{cases} \tag{2}$$

where $\widehat{\mathbf{p}}_i^t$ is the predicted position of the i-th pedestrian at time step t. $\mathbf{e}_i^t$ is the spatial representation of the i-th pedestrian's trajectory at time step t. The embedding function is defined as:

$$\phi\left(\mathbf{x}; W\right) \triangleq W\mathbf{x}. \tag{3}$$

Then, we use a maxpooling operation to combine the spatial representation of all pedestrians, obtaining the spatial representation of the crowd at time step t,

$$\mathbf{s}^t = MaxPooling\left(\mathbf{e}_1^t, \mathbf{e}_2^t, \cdots, \mathbf{e}_N^t\right), \tag{4}$$

Spatial representation $\mathbf{s}^t$ contains information of the crowd at a single time step. However, pedestrians interact with each other dynamically. To improve the accuracy of predictions, a spatio-temporal representation is required.

2) Spatio-temporal representation: In the second step, the hub network feeds a set of spatial representations $\left\{\mathbf{s}^1, \mathbf{s}^2, \cdots, \mathbf{s}^{T_{obs}}\right\}$ of sequential time steps into a LSTM. Then, the LSTM combines all the spatial representations in its hidden state. Thus, the hidden state of the LSTM is a spatio-temporal representation $\mathbf{r}^t$ of all pedestrians. Specifically, we can calculate $\mathbf{r}^t$ as follows:

$$\begin{cases} \mathbf{h}_c^0 = \mathbf{0}, \\ \mathbf{e}^t = \phi\left(\mathbf{s}^t; W_2\right), \\ \left[\mathbf{o}_c^t, \mathbf{h}_c^t\right] = LSTM\left(\mathbf{h}_c^{t-1}, \mathbf{e}^t; W_3\right), \\ \mathbf{r}^t = \phi\left(\mathbf{o}_c^t; W_4\right), \end{cases} \tag{5}$$

where W_2 and W_4 are the embedding weights, W_3 is the weight of LSTM. $\mathbf{o}_c^t$ and $\mathbf{h}_c^t$ are the output and hidden state of the LSTM respectively.

Note that, $\mathbf{r}^t$ depends on the observed trajectories of all pedestrians. Hence, our algorithm is able to consider complicated interactions among multiple pedestrians. This property allows our algorithm to produce accurate predictions. Meanwhile, $\mathbf{r}^t$ is able to be obtained in a single forward propagation of the hub network at each time step. In other words, the time complexity of computing interactions among pedestrians is linear to the number of pedestrians N. This property allows our algorithm to be computationally efficient. By contrast, conventional algorithms compute pairwise interactions, leading to oversimplification of the interactions among pedestrians. Also, the number of pairwise interactions increases quadratically as N increases.

C. The host networks

The spatio-temporal representation $\mathbf{r}^t$ is then employed by host networks. For the i-th pedestrian, the host network first embeds the observed trajectory O_i, and then combines the embedded trajectory with the spatio-temporal representation $\mathbf{r}^t$, predicting the future trajectory. Specifically, the host network predicts the future trajectory by two steps.

First, the host network takes the observed trajectory O_i and the spatio-temporal representation $\mathbf{r}^t$ as input and generates an integrated representation $\mathbf{q}_i^t$,

$$\mathbf{q}_i^t = \begin{cases} \mathbf{r}^t \odot \phi\left(\mathbf{p}_i^t; W_5\right), & if\ t \in [1, T_{obs}], \\ \mathbf{r}^t \odot \phi\left(\widehat{\mathbf{p}}_i^t; W_5\right), & if\ t \in \left[T_{obs}+1, T_{obs}+T_{pred}\right], \end{cases} \tag{6}$$

where W_5 is the embedding weight, and $\odot$ denotes the point-wise multiplication. $\mathbf{q}_i^t$ depends on both the trajectory

of the i-th pedestrian and the interactions between the i-th pedestrian and others in the crowd.

Second, the host network predicts the future trajectory of the i-th pedestrian depending on the observed trajectory O_i and the integrated representation $\mathbf{q}_i^t$. To encourage the host network to produce non-deterministic predictions, a random noise $\mathbf{z}$, which is sampled from a Gaussian distribution with mean 0 and variance 1, is concatenated to the input of the host network. Specifically, the host network encodes the observed trajectory O_i with the hidden state $\mathbf{h}_{ei}^{T_{obs}}$, i.e.,

$$\begin{cases} \mathrm{d}\mathbf{p}_i^0 = \mathbf{0}, \\ \mathrm{d}\mathbf{p}_i^{t-1} = \mathbf{p}_i^t - \mathbf{p}_i^{t-1}, \\ [\mathbf{o}_{ei}^t, \mathbf{h}_{ei}^t] = LSTM_E\left(\mathbf{h}_{ei}^{t-1}, [\mathbf{q}_i^t, \mathrm{d}\mathbf{p}_i^{t-1}]; W_6\right), \\ t \in [1, T_{obs}], \end{cases} \tag{7}$$

where $LSTM_E(\cdot)$ with weight W_6 denotes the encoding procedure. Then, the host network proceeds with

$$\begin{cases} [\mathbf{o}_{di}^t, \mathbf{h}_{di}^t] = LSTM_D\left(\mathbf{h}_{di}^{t-1}, [\mathbf{q}_i^t, \mathrm{d}\widehat{\mathbf{p}}_i^{t-1}, \mathbf{z}]; W_7\right), \\ \mathrm{d}\widehat{\mathbf{p}}_i^t = \phi\left(\mathbf{o}_{di}^t; W_8\right), \\ \widehat{\mathbf{p}}_i^t = \widehat{\mathbf{p}}_i^{t-1} + \mathrm{d}\widehat{\mathbf{p}}_i^t, \\ t \in [T_{obs}+1, T_{obs}+T_{pred}], \end{cases} \tag{8}$$

where $LSTM_D(\cdot)$ with weight W_7 is the decoding function. W_8 is the embedding weight of the output layer. And the initial states are set according to,

$$\begin{cases} \mathbf{h}_{di}^{T_{obs}} = \mathbf{h}_{ei}^{T_{obs}}, \\ \widehat{\mathbf{p}}_i^{T_{obs}} = \mathbf{p}_i^{T_{obs}}, \\ \mathrm{d}\widehat{\mathbf{p}}_i^{T_{obs}} = \mathbf{p}_i^{T_{obs}} - \mathbf{p}_i^{T_{obs}-1}. \end{cases} \tag{9}$$

D. Implementation Details

The network configuration of StarNet is detailed in TABLE I.

TABLE I: Network Configuration of AstoridNet

Weight	Weight Dimension
W_1	64x2
W_2	64x64
W_3	64x32, 32x1(bias)
W_4	32x64
W_5	64x2
W_6	64x66, 64x1(bias)
W_7	64x74, 64x1(bias)
W_8	2x64

We train the proposed StarNet with the loss function applied in [13]. Specifically, at the training stage, StarNet produces multiple predicted trajectories for each pedestrian. Each predicted trajectory $\{\widehat{F}_{ik}\}_{k=1}^K$ has a distance to the ground truth trajectory F_i. Only the smallest distance is minimized. Mathematically, the loss function is,

$$L = \frac{1}{NT_{pred}} min_{k=1}^K \sum_{j=1}^{N} \sum_{t=T_{obs}+1}^{T_{obs}+T_{pred}} \left(\widehat{\mathbf{p}}_{jk}^t - \mathbf{p}_j^t\right)^2, \tag{10}$$

where K is the number of sampled trajectories. This loss function improves the training speed and stability. Moreover, we employ an Adam optimizer and set the learning rate to 0.0001.

In practice, all host networks share the same weights, since pedestrians in a scenario have the same behavioral patterns, such as variable-speed movement, sharp turning and so on. In our approach, we use shared weights to learn the aforementioned behavioral patterns. Each host network contains specific LSTM state which captures certain pedestrian's behavior, and predicts the pedestrian's future trajectory. The observed trajectories of all pedestrians form a batch, which is fed into one single implementation of the host network. In this way, the prediction for all pedestrians is able to be obtained in a single forward propagation.

IV. EXPERIMENTS

We evaluate our model on two human crowded trajectory datasets: ETH [24] and UCY [25]. These datasets have 5 sets with 4 different scenes. In these scenes, there exist challenging interactions, such as walking side by side, collision avoidance and changing directions. Following the settings in [11], [13], [14], we train our model on 4 sets and test on the remaining one.

We compare our StarNet with three state-of-the-arts including Social LSTM, Social GAN and Social Attention. Besides, we test the basic LSTM-based encoder-decoder model, which does not consider the interactions among pedestrians, as a baseline.

Following [11], [13], [14], we compare these methods in terms of the Average Displacement Error (ADE) and Final Displacement Error (FDE). The ADE is defined as the mean Euclidean distance between predicted coordinates and the ground truth. Specifically, all methods output 8 coordinates uniformly sampled from the predicted trajectory. Then the distance between such 8 points with the ground truth is accumulated as the ADE. The FDE is the distance between the final point of the predicted trajectory and the final point of the ground truth. All these methods are trained with the loss Eq. (10) to deal with multimodal distribution during evaluation. Besides, we compare the computational time of all these methods. All experiments are conducted on the same computational platform with an NVIDIA Tesla V100 GPU.

A. Experimental Results

1) Accuracy: As shown in TABLE II, StarNet outperforms the others in most cases. A possible explanation is that StarNet considers the collective influence among pedestrians all together to make more accurate predictions. In comparison, other state-of-the-arts only model the pairwise interactions between pedestrians.

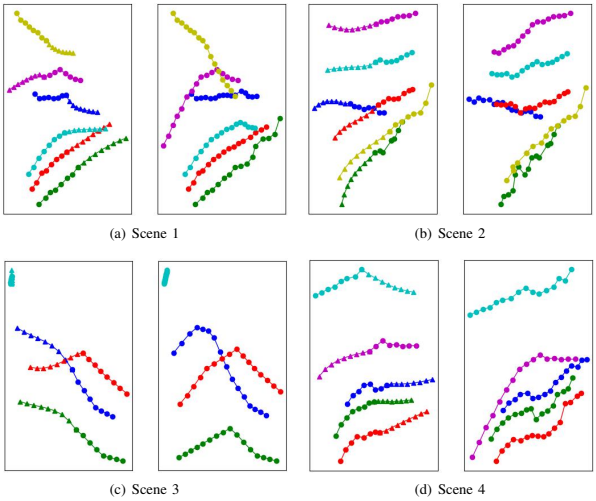
Interestingly, we notice that the test datasets include multiple senses. In these scenes, StarNet has the smallest variances of ADE and FDE, which means that StarNet is robust against the changes of scenes.

TABLE II: Comparison of Prediction Errors

Metric	Dataset	LSTM	Social LSTM	Social GAN	Social Attention	StarNet (Ours)
ADE	ZARA-1	0.25	0.27	**0.21**	1.66	0.25
	ZARA-2	0.31	0.33	0.27	2.30	**0.26**
	UNIV	0.36	0.41	0.36	2.92	**0.21**
	ETH	0.70	0.73	0.61	2.45	**0.31**
	HOTEL	0.55	0.49	0.48	2.19	**0.46**
Average ADE	-	0.43	0.45	0.39	2.30	**0.30**
Variance of ADE	-	0.028	0.026	0.021	0.166	**0.008**
FDE	ZARA-1	0.53	0.56	**0.42**	2.64	0.47
	ZARA-2	0.65	0.70	0.54	4.75	**0.53**
	UNIV	0.77	0.84	0.75	5.95	**0.40**
	ETH	1.45	1.48	1.22	5.78	**0.54**
	HOTEL	1.17	1.01	0.95	4.94	**0.91**
Average FDE	-	0.91	0.91	0.78	4.81	**0.57**
Variance of FDE	-	0.118	0.101	0.802	1.394	**0.031**

TABLE III: Comparison of Computational Time

Metric	LSTM	Social LSTM	Social GAN	Social Attention	StarNet (Ours)
Inference Time (Seconds)	**0.029**	0.504	0.202	3.714	**0.073**
Number of Paramters (Kilo)	**22.87**	156.06	108.03	874.95	**31.90**



(a) Scene 1 (b) Scene 2

(c) Scene 3 (d) Scene 4

Fig. 3: Predicted trajectories and the corresponding ground truths. Different colors indicate different trajectories. The trajectories of ground truth are labeled with dots. The predicted trajectories are labeled with triangles.

To assess StarNet qualitatively, we illustrate the prediction results in 4 scenes, as shown in Figure 3. In each scene, the left sub-figure presents the observed trajectories and the predicted trajectories of all pedestrians. The right sub-figure shows the trajectories of ground truth.

We can observe that StarNet could handle complicated

interactions among pedestrians. Most predicted trajectories accurately reflect the pedestrians' movements and have no collisions with other trajectories. However, there are some failure cases due to the multimodal distribution of future trajectories. For example, in 3(c), the predictions for the blue and green trajectories fail to match the ground truth. We argue that although these predicted trajectories do not match the ground truth, these trajectories are still plausible in crowded scenes.

2) Computational time cost: When deployed in mobile robots and autonomous vehicles, the prediction algorithm needs to be invoked with a high frequency. Hence the computational time of the prediction algorithm is a crucial property.

As shown in TABLE III, the basic LSTM model is the fastest model since the model takes no interactions among pedestrians into consideration. StarNet is the second fastest model. Specifically, StarNet is 51 times faster than Social Attention, 7 times faster than Social LSTM, and 3 times faster than Social GAN. Meanwhile, the number of parameters employed by StarNet is less than state-of-the-arts by a large margin. StarNet is computationally efficient since the interpersonal interactions among pedestrians are computed in a single forward propagation, as discussed in Section II.

V. CONCLUSION

In this paper, we propose StarNet, which has a star topology, for pedestrian trajectory prediction. StarNet learns complicated interpersonal interactions and predicts future trajectories with low time complexity. We apply a centralized hub network to model the spatio-temporal interactions among pedestrians. Then the host network takes full advantage of the spatio-temporal representation and predicts pedestrians' trajectories. We demonstrate that StarNet outperforms state-of-the-arts in multiple experiments.

REFERENCES

[1] D. Ferguson, D. Michael, U. Chris and K. Sascha, "Detection, prediction, and avoidance of dynamic obstacles in urban environments," *in 2008 IEEE International Conference on Intelligent Vehicles Symposium (IVS)*. IEEE, 2008, pp. 1149-1154.

[2] Y. Luo, P. Cai, A. Bera, D. Hsu, W. S. Lee and D. Manocha, "Porca: Modeling and planning for autonomous driving among many pedestrians," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3418-3425, 2018.

[3] F. Large, D. Vasquez, T. Fraichard and C. Laugier, "Avoiding cars and pedestrians using velocity obstacles and motion prediction," *in 2004 IEEE International Conference on Intelligent Vehicles Symposium (IVS)*. IEEE, 2004, pp. 375-379.

[4] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey and S. Srinivasa, "Planning-based prediction for pedestrians," *in 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2009, pp. 3931-3936.

[5] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," *in 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2010, pp. 797-803.

[6] N. E. D. Toit and J. W. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 101-115, 2012.

[7] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, pp. 4282, 1995.

[8] S. Yi, H. Li and X. Wang, "Understanding pedestrian behaviors from stationary crowd groups," *in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 3488-3496.

[9] S. Yi, H. Li and X. Wang, "Pedestrian behavior modeling from stationary crowds with applications to intelligent surveillance," *IEEE transactions on image processing*, vol. 25, no. 9, pp. 4354-4368, 2016.

[10] B. Zhou, X. Wang and X. Tang, "Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents," *in 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 2871-2878.

[11] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, F. Li and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," *in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE 2016, pp. 961-971.

[12] H. Wu, Z. Chen, W. Sun, B. Zheng and W. Wang, "Modeling trajectories with recurrent neural networks," *in 28th International Joint Conference on Artificial Intelligence (IJCAI)*. 2017, pp. 3083-3090.

[13] A. Gupta, J. Johnson, F. Li, S. Savarese and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks," *in 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018, pp. 2255-2264.

[14] A. Vemula, K. Muelling and J. Oh, "Social attention: Modeling attention in human crowds," *in 2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1-7.

[15] Y. Xu, Z. Piao and S. Gao S, "Encoding crowd interaction with deep neural network for pPedestrian trajectory prediction," *in 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018, pp. 5275-5284.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial nets," *in 28th Conference on Neural Information Processing Systems (NIPS)*. 2014, pp. 2672-2680.

[17] C. M. Bishop, "Mixture density networks," *Technical Report NCRG/4288*, Aston University, Birmingham, UK, 1994.

[18] D. Ha and D. Eck, "A neural representation of sketch drawings," *arXiv preprint* arXiv:1704.03477, 2017.

[19] E. Schmerling, K. Leung, W. Vollprecht and M. Pavone, "Multimodal probabilistic model-based planning for human-robot interaction," *in 2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1-9.

[20] K. Cho, B. V. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint* arXiv:1406.1078, 2014.

[21] K. Cho, B. V. Merrienboer, D. Bahdanau and Y. Bengoi, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint* arXiv:1409.1259, 2014.

[22] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," *in 2015 International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4945-4949.

[23] C. R. Qi, H. Su, K and J. G. Leonidas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *in 2017 Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017, pp. 652-660.

[24] S. Pellegrini, A. Ess, K. Schindler and L. V. Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," *in 2009 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2009, pp. 261-268.

[25] A. Lerner, Y. Chrysanthou and D. Lischinski, "Crowds by example," *Computer Graphics Forum*, vol. 26, no. 3, pp. 655-664, 2007.

[26] D. Varshneya, G. Srinivasaraghavan, "Human trajectory prediction using spatially aware deep attention models," *arXiv preprint* arXiv:1705.09436, 2017.

[27] T. Fernando, S. Denma, S. Sridharan and C. Fookes, "Soft+hardwired attention: An lstm framework for human trajectory prediction and abnormal event detection," *arXiv preprint* arXiv:1702.05552, 2017.

2019 INFORMS Annual Meeting

2019 INFORMS Annual Meeting
Seattle, WA, October 20-23, 2019
| ISBN 978-0-9906153-1-6
https://doi.org/10.1287/infp.2019.XXXX
pp. 000–000

A Two-Stage Fast Heuristic for Food Delivery Route Planning Problem

Huanyu Zheng, Shengyao Wang, Ying Cha, Feng Guo, Jinghua Hao, Renqing He, Zhizhao Sun
Meituan-Dianping Group, No. 4 Wangjing East Street, Chaoyang District, Beijing, China,
{zhenghuanyu, wangshengyao, chaying, guofeng13, haojinghua, herenqing,
sunzhizhao}@meituan.com

Abstract Online food-delivery platforms are expanding choice, allowing customers to order from a wide variety of restaurants. As an industrial level technology, route planning algorithm is required to be fast enough. This paper proposes a two-stage fast heuristic for route planning, which solves the problem at millisecond level. To speed up the algorithm, we further utilize geographic information so that invalid search attempts are prevented. Finally, we compare our algorithm with brute-force algorithm and several state-of-the-art algorithms to show its effectiveness and efficiency.

Keywords food delivery, pickup and delivery, time windows, heuristics

1. Introduction

"Tap, order, and eat, all at home." Food ordering and delivery is a fast growing market all over the world. Worldwide, food ordering and delivery companies, such as Grubhub, Uber Eats, and Just Eat, are developing a $94 billion online food ordering and delivery business [4]. In China, over 300 billion customers order food from Meituan-Dianping food delivery platform with more than 3.6 million restaurants to choose, exceeding 24 million daily orders and deliveries in 2018 [9].

As shown in FIGURE 1, after a customer orders a meal, the food delivery platform pushes the order to a restaurant. At the same time, the platform dispatches this order to a driver, and plans a route for him. The dispatching system is shown in FIGURE 2. A real route is shown in FIGURE 3, in which the driver is planned to pick up four meals before delivering two of the meals to corresponding customers. Then, the driver pick up the last meal and deliver the rest meals to customers. At rush hours like lunch time, a driver may run with 10 orders for example, which means that he has to scurry across 10 restaurants and 10 customers. Under this setting, there are 2.38×10^{15} ways of route planning, which is hard to solve in limited time.

In this paper, we introduce an algorithm to find a satisfactory route, aiming to minimize delays and the route length. The algorithm is required to be highly efficient because it is an essential part of dispatching algorithm. Different from traditional logistics, food delivery dispatching algorithm is an online algorithm, which matches thousands of orders with thousands of drivers each time. Thus, route planning algorithm has to plan a route within milliseconds.

FIGURE 1. Food ordering and delivery process.

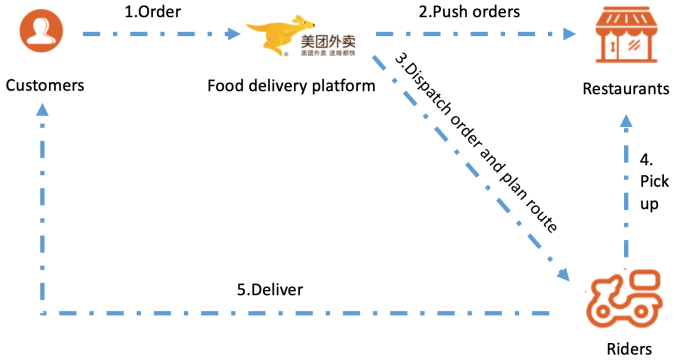


FIGURE 2. Dispatching algorithm.

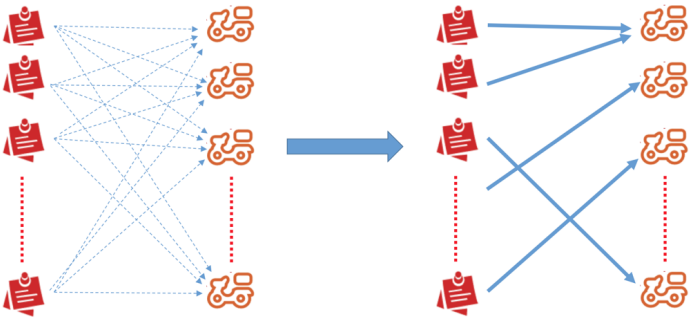
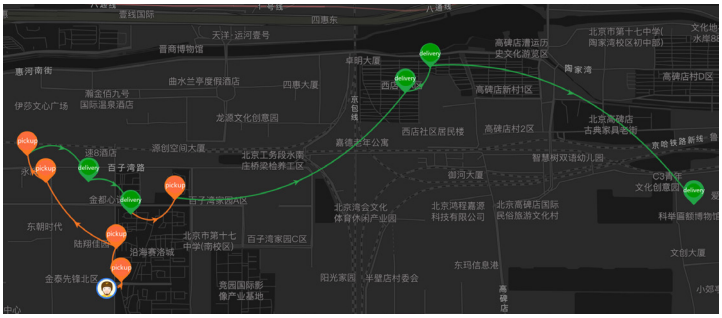


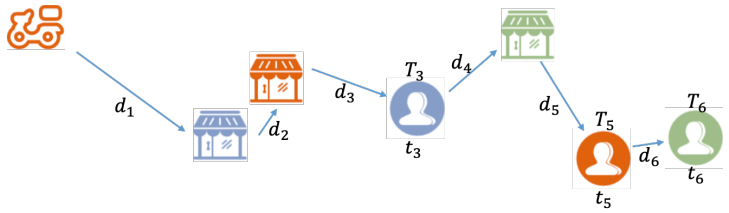
FIGURE 3. A real route.



2. Literature Review

Food delivery route planning problem is rarely studied in the literature. The most relevant literature is the Pickup and Delivery Problem with Time Windows (PDPTW), which models

FIGURE 4. A typical route.



a fleet of vehicles serving a collection of transportation requests. Each request specifies a pickup location and a delivery location. Vehicles are routed to serve all requests, optimizing a certain objective function such as total distance traveled, with precedence constraint and capacity constraint.

Problem in this paper can be modeled as the single vehicle PDPTW, which considers only one vehicle and several customers. [14] propose a variable-depth search algorithm, which produces near optimal solutions most of the time, but may end up with an infeasible solution. Then, they have to spend a large computation time with simulated annealing. [5] present a genetic algorithm, a simulated annealing algorithm and a hill climbing approach. They find simulated annealing algorithm is superior to other algorithms but requires longer running time.

A mathematical formulation of the PDPTW involving a single depot is given in [2]. [13] further formulate a general version of the PDPTW, and present a survey of the problem. In the past two decades, several exact algorithms and heuristics are designed to solve the PDPTW. Exact algorithms include column generation [2], branch-and-cut [7], and branch-and-cut-and-price [11, 1]. Heuristics include tabu search [10], insertion-based heuristic [8], adaptive large neighborhood search [12] and simulated annealing [15].

The PDPTW is applied to problems arising in logistics and public transit, such as transporting goods [2] and home health care [6]. The problems are usually in large scale and are acceptable to be solved in hours. However, food delivery route planning problem demands very low computational complexity. As a core algorithm supporting the food delivery platform, the time complexity of route planning algorithm is limited in only several milliseconds. In other words, algorithms proposed by papers above are not applicable to our problem. In this paper, we propose a new heuristic approach for PDPTW problem to meet this restrict time complexity requirement.

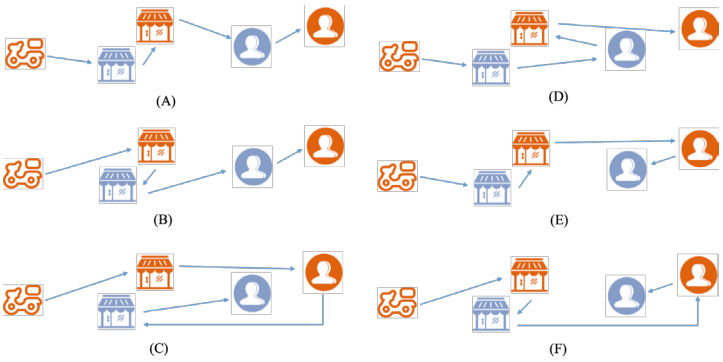
3. Algorithm

In this section, we present our Two-Stage Fast Heuristic (TSFH), including initialization and local search strategies. FIGURE 4 shows a typical route, where t_i is the Estimated Time of Arrival (ETA) [3] of a delivery point, which is shown to customer and restaurant as soon as the order generates. T_i is estimated time of driver's action, calculated by our route planning algorithm, and d_i is traveling distance from previous point to i-th point. Then we have Formula 1 presenting our objective, which minimizes delays and route length.

$$\min \sum_{i=1}^{n} [\max(T_i - t_i, 0) + d_i] \tag{1}$$

The problem has two constraints. (1) precedence constraint: A driver has to pick up meals before delivery. (2) capacity constraint: The total capacity of a driver is limited during the route.

FIGURE 5. Possible ways of insertion.



The TSFH has two stages. Stage I initializes one feasible solution with greedy search mechanism. To speed up the insertion, we utilize geographic knowledge to avoid invalid search. Stage II adjusts the solution with two local search strategies.

3.1. Stage I: Initialization

3.1.1. Initialization with Greedy Insertion The initialization stage can be described as follows. First, we sort orders according to their ETA. Then, we plan the first order. Due to precedence constraint, the only plan is to plan pickup point before delivery point. For the following orders, their pickup and delivery points are inserted into the route according to the objective. For instance, as shown in FIGURE 5, we have 6 ways to insert the second order to the route. We greedily insert pickup point and delivery point to minimize delays and route length. According to this criterion, FIGURE 5(A) is the optimal way to insert. After all points are inserted, we formulate a feasible route.

3.1.2. Speeding up with Geographic Information In China, restaurants are geographically close to each other. For example, restaurants at a central business district building may serve 60% customers within 5 kilometers. Also, customers are geographically close, most customers are gathered in certain communities. Thus, we can cluster pickup and delivery points by hierarchical clustering. With these clusters, we speed up the initialization stage by reducing "bad" insertion attempts.

Clustering algorithm is described as follows, where D is a given range, say 100 meters. For each point i, if it is not classified, it generates a new group and makes itself a center point. For each point j, if it not classified, it is classified into group i if $d_{ij} < D$; if it is classified to group k and is not a center point, then if $d_{ij} < d_{kj}$, we reclassify it into group i.

Lemma 1 (Insertion Before Own Group). *If point j is classified to group i, then inserting point j to groups before group i is worse than inserting point j to group i.*

Proof. Point j is classified to group i. Inserting point j to group k before group i must be worse than directly inserting it to group i, because the route length is longer, but no delivery point benefits from shorter delays.

FIGURE 6 gives an example illustrating the above lemma. Pickup and delivery points are classified into three groups. Consider insertion of 'green' pickup and delivery points, given 'blue' points and 'orange' points inserted beforehand. We can see from FIGURE 6, inserting a point before its own group (FIGURE 6 (B)) is always worse than inserting it to its own group (FIGURE 6 (A)), as the driver travels more and customers may suffer from more delays.

FIGURE 6. Insertion before own group.

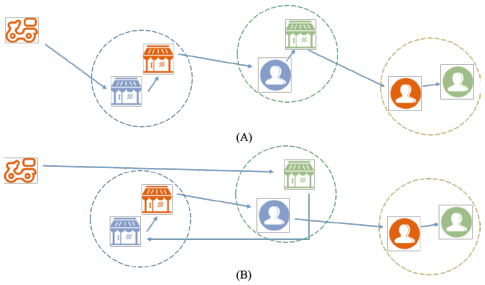
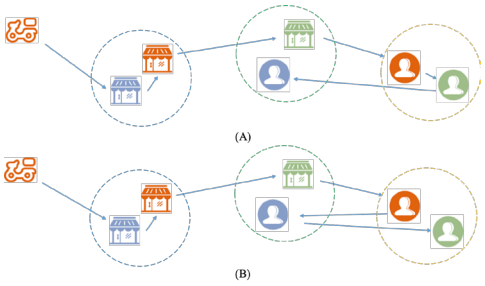


FIGURE 7. Insertion after own group.



Lemma 2 (Insertion After Own Group). *If point j is classified to group i, then we can always find an insertion better than the insertion of point j to group k after group i.*

Proof. Point j is classified to group i. Inserting point j to group k after group i must be worse than inserting it between group k and group $k+1$ (if group k is the last group, then inserting it to the end), because the route length is longer, but no delivery point benefits from shorter delays.

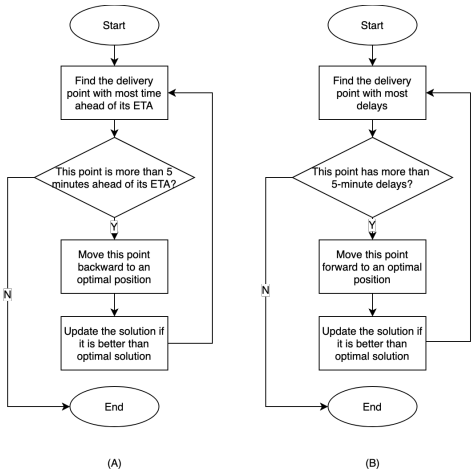
FIGURE 7 gives an example illustrating the above lemma. Pickup and delivery points are classified into three groups. Consider insertion of 'blue' pickup and delivery points, given 'green' points and 'orange' points inserted beforehand. We can see from FIGURE 7, inserting a point between a group after its own group (FIGURE 7 (B)) is always worse than inserting it to the last position (FIGURE 7 (A)), as the driver travels more and customers may suffer from more delays.

From Lemma 1 and Lemma 2, we conclude that point j is only worth to insert when it is inserted into its group, say group i, or between groups after group i. This conclusion reduces invalid insertion attempts and speeds up the algorithm.

3.2. Stage II: Local Search

FIGURE 8 shows the local search stage. After initialization stage, the local search improves the route by looking for better solutions at solution neighborhood. Our algorithm considers two kinds of neighborhood. First, we find delivery points with most delays and move them backward to an optimal position, which is shown in FIGURE 8 (A). Second, we find delivery points with most sufficient time and move them forward to an optimal position, which is shown in FIGURE 8 (B). Each time we find a better solution, the best solution is replaced.

FIGURE 8. Local search.



4. Numerical Results and Experiments

In this section, we provide numerical experiments to compare different algorithms. First, we compare initialization without speeding up technique and fast initialization with speeding up technique to show the effectiveness of initialization stage. Then, we use brute-force search to find the optimal solution to each problem as a baseline. Moreover, we compare our TSFH algorithm with the optimal solution to show our TSFH can generate near optimal solutions in limited time. Moreover, the TSFH is compared with several state-of-the-art algorithms to show its effectiveness and efficiency.

Instances are randomly sampled from real route planning problems. We partition the instance set into three sets, '< 10', '10 to 20', and '> 20', according to the number of pickup and delivery points.

In the following tables, we show performance of algorithms by total score and average time. Total score is the sum of delays (in minutes) and route length (in kilometers). Average time is the computation time of one instance (in milliseconds).

All experiments are run on a MacBook Pro with 2.2 GHz processors / 16 GB RAM in Mac-OS. The algorithms are coded in Java using Eclipse.

4.1. Comparison of Initialization and Fast Initialization

To show the effectiveness of initialization and local search, and efficiency of speeding up technique, we compare initialization and fast initialization using geographic information.

From TABLE 1, we can see that initialization can be sped up considering geographic information. For example, the average running time of the '10 to 20' instance set is reduced from 0.37ms to 0.21ms, which is 43.2% faster. Moreover, effectiveness of initialization is not harmed by speeding up as total score is nearly the same. Therefore, we have shown that our algorithm can generate near optimal solutions within several milliseconds.

4.2. Optimal Solutions by Brute-force Algorithm

The optimal solution can be solved using a brute-force search algorithm. Consider a driver with n pickup tasks and n delivery tasks, the complexity of brute-force search algorithm can be derived as follows. A solution from $2n$ points' full permutation can only be feasible when

TABLE 1. Numerical results of initialization and fast initialization.

Instance	Algorithm	Total Score	Delays	Route Length	Average Time
> 20	Initialization	240.88	232.17	8.71	2.60
> 20	Fast Initialization	242.48	233.74	8.73	1.08
10 to 20	Initialization	46.29	41.35	4.94	0.37
10 to 20	Fast Initialization	46.11	41.19	4.93	0.21
< 10	Initialization	28.59	24.09	4.50	0.14
< 10	Fast Initialization	28.67	24.18	4.50	0.10

TABLE 2. Numerical results of our TSFH and brute-force algorithm.

Instance	Algorithm	Total Score	Delays	Route Length	Average Time
> 20	TSFH	223.30	214.79	8.51	7.08
10 to 20	TSFH	42.48	37.65	4.83	1.01
< 10	TSFH	27.35	22.94	4.40	0.41
< 10	Brute-force	27.17	22.79	4.38	199.31

TABLE 3. Numerical results of our TSFH and the state-of-the-art algorithms.

Instance	Algorithm	Total Score	Delays	Route Length	Average Time
> 20	TSFH	**223.30**	214.79	8.51	**7.08**
> 20	VDS [14]	877.80	869.63	8.18	25.96
> 20	SA [5]	1195.50	1185.74	9.75	64.71
10 to 20	TSFH	**42.48**	37.65	4.83	**1.01**
10 to 20	VDS [14]	77.34	72.66	4.68	6.23
10 to 20	SA [5]	101.02	96.18	4.84	35.59
< 10	TSFH	**27.35**	22.94	4.40	**0.41**
< 10	VDS [14]	35.17	30.95	4.23	3.51
< 10	SA [5]	36.87	32.62	4.24	25.64
< 10	Brute-force	27.17	22.79	4.38	199.31

each pair of pickup and delivery points follows precedence constraint. Thus, the complexity is $O\left(\frac{(2n)!}{2^n}\right)$, if capacity constraint is not considered. As the complexity grows dramatically with the number of pickup or delivery points, we only provide results of '< 10' instance sets.

4.3. Comparison of Our TSFH and Brute-force Algorithm

In this subsection, we compare the TSFH with brute-force algorithm, which generates optimal solutions to each instance. TABLE 2 shows the results. From '< 10' instance set, we can find that our full algorithm uses only 0.21% of average time but generates nearly the same performance than brute-force algorithm. We can also see that the average computation time of the TSFH is limited in milliseconds in '10 to 20' and '< 10' instances. As most instances generated in real life have less than 20 pickup and delivery points, we show the TSFH is efficient enough to solve food delivery route planning problem within several milliseconds.

4.4. Comparison of Our TSFH and the State-of-the-art Algorithms

In this subsection, we compare the best results of our TSFH to those of variable-depth search (VDS) [14] and simulated annealing (SA) [5]. The comparative results are listed in TABLE 3. We can see that the TSFH outperforms VDS and SA in terms of total score and average running time. Particularly, in '< 10' instances, the TSFH reaches near optimal solutions, which is 0.7% above optimal, while VDS and SA are 29.4% and 35.7% above optimal respectively. Moreover, the TSFH also solves '< 10' instances in 0.41ms per instance, while VDS and SA solve those instances in 3.51ms and 25.64ms, respectively. Real life food delivery route planning demands high running speed. The TSFH can solve all '< 10' instances in 1 millisecond, while the computation time of VDS and SA is not acceptable. Therefore, we conclude that the TSFH is more effective and efficient than the state-of-art algorithms, and more suitable to be applied to real life problem solving.

5. Conclusions

In this paper, food delivery route planning problem is modeled as the single vehicle pickup and delivery problem with time windows (PDPTW). Unlike traditional approaches solving the single vehicle PDPTW, our paper presents a two-stage fast heuristic (TSFH). In stage I, a feasible and near optimal solution is generated in a greedy heuristic way. With restaurants and customers location cluster information, our algorithm is accelerated by avoiding bad searching attempts. In stage II, we improve the solution in stage I by exploiting two neighborhoods. From our numerical results, we represent the effectiveness and efficiency by comparing our results with those from brute-force algorithm and some best algorithms in the literature. Our TSFH algorithm generates near optimal solutions within milliseconds.

References

[1] Baldacci R, Bartolini E, Mingozzi A (2011) An exact algorithm for the pickup and delivery problem with time windows. *Operations Research* 59(2):414–426.

[2] Dumas Y, Desrosiers J, Soumis F (1991) The pickup and delivery problem with time windows. *European Journal of Operational Research* 54(1):7–22.

[3] Fang Z, Huang L, Wierman A (2017) Prices and subsidies in the sharing economy. *Proceedings of the 26th International Conference on World Wide Web*, 53–62 (International World Wide Web Conferences Steering Committee).

[4] Hirschberg C, Rajko A, Schumacher T, Wrulich M (2016) The changing market for food delivery. https://www.mckinsey.com/industries/high-tech/our-insights/the-changing-market-for-food-delivery.

[5] Hosny MI, Mumford CL (2010) The single vehicle pickup and delivery problem with time windows: intelligent operators for heuristic and metaheuristic algorithms. *Journal of Heuristics* 16(3):417–439.

[6] Liu R, Xie X, Augusto V, Rodriguez C (2013) Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care. *European Journal of Operational Research* 230(3):475–486.

[7] Lu Q, Dessouky MM (2004) An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science* 38(4):503–514.

[8] Lu Q, Dessouky MM (2006) A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. *European Journal of Operational Research* 175(2):672–687.

[9] Meituan-Dianping Group (2019) Meituan to invest rmb 11 billion to support merchant development. https://www.prnewswire.com/news-releases/meituan-to-invest-rmb11-billion-to-support-merchant-development-300782802.html.

[10] Nanry WP, Barnes JW (2000) Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological* 34(2):107–121.

[11] Ropke S, Cordeau JF (2009) Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science* 43(3):267–286.

[12] Ropke S, Pisinger D (2006) An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science* 40(4):455–472.

[13] Savelsbergh MW, Sol M (1995) The general pickup and delivery problem. *Transportation Science* 29(1):17–29.

[14] Van der Bruggen L, Lenstra JK, Schuur P (1993) Variable-depth search for the single-vehicle pickup and delivery problem with time windows. *Transportation Science* 27(3):298–311.

[15] Wang C, Mu D, Zhao F, Sutherland JW (2015) A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows. *Computers & Industrial Engineering* 83:111–122.

2019 International Joint Conference on Artificial Intelligence

Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)

Earlier Attention? Aspect-Aware LSTM for Aspect-Based Sentiment Analysis

Bowen Xing[1*], **Lejian Liao**[1], **Dandan Song**[1 †], **Jingang Wang**[2], **Fuzhen Zhang**[2], **Zhongyuan Wang**[2] and **Heyan Huang**[1]
[1]Lab of High Volume language Information Processing & Cloud Computing
Beijing Lab of Intelligent Information Technology
School of Computer Science & Technology, Beijing Institute of Technology
[2]Meituan-Dianping Group
{xingbowen,liaolj,sdd,hhy63}@bit.edu.cn,
{wangjingang02,zhangfuzheng,wangzhongyuan02}@meituan.com

Abstract

Aspect-based sentiment analysis (ABSA) aims to predict fine-grained sentiments of comments with respect to given aspect terms or categories. In previous ABSA methods, the importance of aspect has been realized and verified. Most existing LSTM-based models take aspect into account via the attention mechanism, where the attention weights are calculated after the context is modeled in the form of contextual vectors. However, aspect-related information may be already discarded and aspect-irrelevant information may be retained in classic LSTM cells in the context modeling process, which can be improved to generate more effective context representations. This paper proposes a novel variant of LSTM, termed as aspect-aware LSTM (AA-LSTM), which incorporates aspect information into LSTM cells in the context modeling stage before the attention mechanism. Therefore, our AA-LSTM can dynamically produce aspect-aware contextual representations. We experiment with several representative LSTM-based models by replacing the classic LSTM cells with the AA-LSTM cells. Experimental results on SemEval-2014 Datasets demonstrate the effectiveness of AA-LSTM.

1 Introduction

With increasing numbers of comments on the Internet, sentiment analysis is attracting interests from both research and industry. Aspect-based sentiment analysis is a fundamental and challenging task in sentiment analysis, which aims to infer the sentiment polarity of sentences with respect to given aspects. For example, *"Great salad but the soup tastes bad"*. It's obvious that the opinion over the *'salad'* is positive while the opinion over the *'soup'* is negative. In this case, aspects are included in the comments, and predicting aspect sentiment polarities of this kind of comments is termed aspect term sentiment analysis (ATSA) or target sentiment analysis (TSA). There is another case where the aspect is not explicitly included in the comment. For example, *"Although the dinner is expensive, waiters are so warm-hearted!"*. We can observe that there are two aspect categories mentioned in this comment: price and service with completely opposite sentiment polarities. Predicting aspect sentiment polarities of this kind of comments is termed aspect category sentiment analysis (ACSA), and the aspect categories usually belong to a predefined set. In this paper, we collectively refer aspect category, aspect term/target as aspect. And our goal is aspect-based sentiment analysis (ABSA) including ATSA and ACSA.

As deep learning have been successfully exploited in various NLP tasks [Zhen *et al.*, 2017; Yang and Mitchell, 2017; Xu *et al.*, 2017; Devamanyu *et al.*, 2018], many neural networks have been applied to ABSA. With the ability of handling long-term dependencies, Long Short-Term Memory neural network (LSTM) [Hochreiter and Schmidhuber, 1997] is widely used for context modeling in ABSA, and many recent best performing ABSA methods are based on LSTM because of its significant performance [Tang *et al.*, 2016a; Wang *et al.*, 2016; Chen *et al.*, 2017; Ma *et al.*, 2017; Devamanyu *et al.*, 2018; Xin *et al.*, 2018]. Current mainstream LSTM-based ABSA models adopt LSTM to model the context, obtaining hidden state vectors for each token in the input sequence. After obtaining contextual vector representations, they utilize attention mechanism to produce the attention weight vector.

Recent well performing LSTM-based ABSA models can be divided into three categories according to their way of modeling context: (1) "Attention-based LSTM with aspect embedding (ATAE-LSTM)" [Wang *et al.*, 2016] and "modeling inter-aspect dependencies by LSTM (IAD-LSTM)" [Devamanyu *et al.*, 2018] model the context and aspect together via concatenating the aspect vector to the word embeddings of context words in the embedding layer. (2) "Interactive attention networks (IAN)" [Ma *et al.*, 2017] and "aspect fusion LSTM (AF-LSTM)" [Tay *et al.*, 2018] model the context alone and utilize the aspect to compute context's attention vector in the attention mechanism. (3) "Recurrent attention network on memory (RAM)" [Chen *et al.*, 2017] introduces relative position information of context words and the given target into their hidden state vectors.

[*]This work was partially done during Bowen's internship at Meituan-Dianping Group.
[†]Corresponding author.

The first category conducts simple joint modeling of contexts and aspects. In the secondary category, on behalf of most of the existing LSTM-based methods, the models only use context words as input when modeling the context, so they get the same context hidden states vectors when analysing comments containing multiple aspects. The second category models the context separately while utilizing the aspect information in context's attention calculation. And the method in the third category additionally multiplies a relative position weight. However, no aspect information is considered in the LSTM cells of all the above methods.Therefore, after context modeling, their hidden state vectors contain the information that is important to the "overall" comment semantics. This is determined by the functionality of classic LSTM. It retains important information and filters out useless information at the sentence-level semantic, in the hidden states corresponding to every context word.

In contrast, for the aspect-based sentiment analysis task, we think the context modeling should be aspect-aware. For a specific aspect, on one hand, some of the semantic information of the whole sentence is useless. These aspect irrelevant information would adversely harm the final sentiment representation, especially in the situation where multiple aspects exist in one comment. This is because when LSTM encounters an important token for the overall sentence semantics, this token's information is retained in every follow-up hidden state. Consequently, even if a good attention vector is produced via the attention mechanism, these hidden state vectors also contain useless information which is magnified to some extent. On the other hand, information that is important to the aspect may be not sufficiently kept in hidden states because of their small contribution to the overall semantic information of the sentence.

We take two typical examples to illustrate the two issues. First, *"The salad is so delicious but the soup is unsatisfied."*. There are two aspects (*salad* and *soup*) of opposite sentiment polarity. When judging the sentiment polarity of the *'soup'*, the word 'delicious' which modifies *'salad'* is also important to the sentence-level semantics of the whole comment, and its information is preserved in the hidden states vectors of subsequent context words, including 'unsatisfied'. So even if 'unsatisfied' is assigned a large weight in the attention vector, the information of 'delicious' will still be integrated into the final context representation and enlarged. Second, *"Pizza is wonderful compared to the last time we enjoyed at another place, and the beef is not bad, by the way."* Obviously, this sentence is mainly about pizza so classic LSTM will retain a lot of information that modifies *'pizza'* when modeling context. But when judging the polarity of beef, because traditional LSTM does not know the aspect is *'beef'*, much-retained *'pizza'* information causes that the information of *'beef'* is not valued enough in hidden state vectors. We define the above issues as the aspect-unaware problem in the context modeling process of current methods. To the best of our knowledge, this is the first time to propose this problem.

In this paper, we propose a novel LSTM variant termed aspect-aware LSTM (AA-LSTM) to introduce the aspect into context modeling process. In every time step, on one hand, the aspect vector can select key information in the context according to the aspect and keep the important information in context words' hidden states. On the other hand, the vector formed aspect information can influent the process of context modeling and filter useless information for the given aspect. So AA-LSTM can generate more effective context hidden states based on the given aspect. This can be seen as an earlier attention operation on context words. It is worth mentioning that though our AA-LSTM model takes the aspect as input, it does not actually fuse the aspect vector into the representation of the context, but only utilize the aspect to influence the process of modeling context via controlling information flow.

The main contributions of our work can be summarized as follows:

- We propose a novel LSTM variant termed as aspect-aware LSTM (AA-LSTM) to introduce the aspect into the process of modeling context.
- Considering that the aspect is the core information in this task, we fully exploit its potential by introducing it into the LSTM cells. We design three aspect gates to introduce the aspect into the input gate, forget gate and output gate in classic LSTM. AA-LSTM can utilize aspect to improve the information flow and then generate more effective aspect-specific context representation.
- We apply our proposed AA-LSTM to several representative LSTM-based models, and the experimental results on the benchmark datasets demonstrate the validity and generalization of our proposed AA-LSTM.

2 Related Work

In this section, we survey some representative studies in the aspect-based sentiment analysis (ABSA). ABSA is the task of predicting the sentiment polarity of a comment with respect to a set of aspects terms or categories included in the context. The biggest challenge faced by ABSA is how to effectively represent the aspect-specific sentiment information of the comment [Ma *et al.*, 2018]. Although some traditional methods for target sentiment analysis also achieve promising results, they are labor intensive because they have mostly focused on feature engineering or massive extra linguistic resources [Kiritchenko *et al.*, 2014; Wagner *et al.*, 2014].

As deep learning achieved breakthrough success in representation learning, many recent works utilized deep neural networks to automatically extract features and generate the context embedding which is a dense vector formed representation of the comment.

Since the attention mechanism was first introduced to the NLP field [Bahdanau *et al.*, 2014], many sequence-based approaches utilize it to generate more aspect-specific final representations. Attention mechanism in ABSA takes aspect information (usually aspect embedding) and the hidden states of every context word (generated by context modeling) as input and produces a probability distribution in which important parts of the context will be assigned bigger weights according to the aspect information.

There are some CNN-based [Xue and Li, 2018] and memory networks (MNs)-based models for context modeling [Tang *et al.*, 2016b; Tay *et al.*, 2017; Wang *et al.*, 2018]. [Tay *et*

al., 2017] model dyadic interactions between aspect and sentence using neural tensor layers and associative layers with rich compositional operators. [Wang *et al.*, 2018] argue that for the case where several sentences are the same except for different targets, relying attention mechanism alone is insufficient. It designed several memory networks having their own characters to solve the problem.

In particular, LSTM networks are widely used in context modeling because of its advantages for sequence modeling [Tang *et al.*, 2016a; Ma *et al.*, 2017; Devamanyu *et al.*, 2018; Wang *et al.*, 2016; Tay *et al.*, 2018; Ma *et al.*, 2018; Liu and Zhang, 2017; Yang *et al.*, 2017]. We divide recent well-performing methods into three categories according to the process of modeling context: First, modeling the context and aspect via concatenating the aspect vector to the word embeddings of context words in the embedding layer. [Wang *et al.*, 2016] firstly propose aspect embedding, and their ATAE-LSTM learns to attend to different parts of the context according to the aspect embedding. Although IAD-LSTM [Devamanyu *et al.*, 2018] model inter-dependencies between multiple aspects of one comment through LSTM after getting the final representation of the context, it is consistent with ATAE-LSTM [Wang *et al.*, 2016] in the way of context modeling.

Second, modeling the context alone and utilizing the aspect to compute context's attention vector in the attention mechanism. The main difference among this category of models is the calculation method of the attention mechanism. [Ma *et al.*, 2017] propose an interactive attention network (IAN) which models targets and contexts separately. Then it learns the interactions between the context and target in attention mechanism utilizing the averages of context's hidden states and target's hidden states. [Tay *et al.*, 2018] propose Aspect Fusion LSTM (AF-LSTM) model with a novel association layer after LSTM to model word-aspect relation utilizing circular convolution and circular correlation.

Third, introducing relative position information of the given target and context words to the hidden state vectors of context words. RAM [Chen *et al.*, 2017] realizes that the hidden state vector of a word will be assigned a larger weight if it is closer to the target through a relative location vector. This operation is conducted before their recurrent attention layer consisting of GRU cells.

Unlike all the above methods, we propose to introduce he aspect information into the process of context modeling. Our proposed AA-LSTM introduces the aspect into the LSTM cells to control information flow. AA-LSTM can not only select key information in the context according to the aspect and keep the important information in context words' hidden state vectors, but also filter useless information for the given aspect. Then AA-LSTM can generate more effective aspect-specific context hidden state vectors.

3 Aspect-Aware LSTM

In this section we describe our proposed aspect-aware LSTM (AA-LSTM) in detail. Classic LSTM contains three gates (input gate, forget gate and output gate) to control the information flow. We argue that aspect information should be considered into LSTM cells to improve the information flow. It is intuitive that in every time step the degree that aspect is integrated into the three gates of classic LSTM should be different. Therefore, we incorporate aspect vector into classic LSTM cells and design three aspect gates to control how much the aspect vector is imported into the input gate, forget gate and output gate respectively. In this way, we can utilize the previous hidden state and the aspect itself to control how much the aspect is imported in the three gates of classic LSTM. Figure 1 illustrates the architecture of the AA-LSTM network and it can be formalized as follows:

$$a_i = \sigma(W_{ai}\,[A, h_{t-1}] + b_{ai}) \tag{1}$$
$$I_t = \sigma(W_I\,[x_t, h_{t-1}] + a_i \odot A + b_I) \tag{2}$$
$$a_f = \sigma(W_{af}\,[A, h_{t-1}] + b_{af}) \tag{3}$$
$$f_t = \sigma(W_f\,[x_t, h_{t-1}] + a_f \odot A + b_f) \tag{4}$$
$$\widetilde{C_t} = tanh(W_C\,[x_i, h_{t-1}] + b_C) \tag{5}$$
$$C_t = f_t \odot C_{t-1} + I_t \odot \widetilde{C_t} \tag{6}$$
$$a_o = \sigma(W_{ao}\,[A, h_{t-1}] + b_{ao}) \tag{7}$$
$$o_t = \sigma(W_o\,[x_t, h_{t-1}] + a_o \odot A + b_o) \tag{8}$$
$$h_t = o_t * tanh(C_t) \tag{9}$$

where x_t represents the input embedding vector of the context word corresponding to time step t, A stands for the aspect vector, h_{t-1} is previous hidden state, h_t is the hidden state of this time step, σ and $tanh$ are sigmoid and hyperbolic tangent functions, $\odot$ stands for element-wise multiplication, W_{ai}, W_{af}, $W_{ao} \in R^{da\times(dc+da)}$ and W_I, W_f, W_C, $W_o \in R^{dc\times 2dc}$ are the weighted matrices, $b_{ai}, b_{af}, b_{ao} \in R^{da}, b_I, b_f, b_C, b_o \in R^{dc}$ are biases and da, dc stand for the aspect vector's dimension and the number of hidden cells at AA-LSTM respectively. $I_t, f_t, o_t \in R^{dc}$ stand for the input gate, forget gate and output gate respectively. The input gate controls the extent of updating the information from the current input. The forget gate is responsible for selecting some information from last cell state. The output gate controls how much the information in current cell state is output to be the hidden state vector of this time step. Similarly, $a_i, a_f, a_o \in R^{da}$ stand for the aspect-input gate, aspect-forget gate and aspect-output gate respectively. The three aspect-based gates determine the extent of integrating the aspect information into the input gate, forget gate and output gate.

Our proposed AA-LSTM takes two strands of inputs: context word embeddings and the aspect vector. At each time step, the context word entering the AA-LSTM dynamically varies according to the sequence of words in the sentence, while the aspect vector is identical. Specifically, aspect vector is the representation of the target in TSA, and it is the aspect embedding in ACSA. Next, we describe the different components of our proposed AA-LSTM in detail.

3.1 Input Gates

The input gate I_t controls how much new information can be transferred to the cell state. While the aspect-input gate a_i controls how much the aspect is transferred to the input gate I_t. The difference between the AA-LSTM and the classical LSTM lies in the weighted aspect vector input of I_t. The

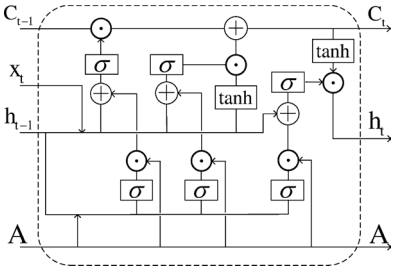


Figure 1: The overall architecture of AA-LSTM network

weight of aspect vector a_i is computed by h_{t-1} and A. h_{t-1} can be regarded as the previous semantic representation of the partial sentences which has been processed in the past time steps. Hence, the extent of the aspect's integration into I_t is decided by the previous semantic representation and the aspect vector A.

3.2 Forget Gates

The forget gate f_t abandons trivial information and retains key information from last cell state C_{t-1}. Similarly, the aspect-input gate a_f controls how much the aspect vector is transferred to the forget gate f_t. The difference between the AA-LSTM and the classical LSTM in f_t is the introduction of weighted aspect vector. And the weight of aspect vector a_f is computed by h_{t-1} and A. Therefore, the extent of the aspect's integration into I_t is decided by the previous semantic representation and the aspect vector A.

3.3 Candidate Cell and Current Cell

The candidate cell $\widetilde{C_t}$ represents the alternative input content. The current cell C_t updates its cell state by selecting important information from last cell state C_{t-1} and $\widetilde{C_t}$.

From Equation 6 we can observe that the alternative input content $\widetilde{C_t}$ aspects two strands of inputs: the last hidden state h_{t-1} and the input embedding x_t of this time step. So the hidden states in each time step contain the semantic information of the previous sentence segment. In the classical LSTM, information that is more important to the overall sentence semantics is more likely to be preserved in the hidden states vectors of the subsequent time steps. However, for some information which is retained because of its contribution to the semantics of the whole sentence, it may be noisy when judging the sentiment polarity of a given aspect. And the information that is crucial to analyze the given aspect's sentiment may be neglected due to its less contribution to the overall sentence. As demonstrated in the Introduction section, we define this phenomenon as an aspect-unaware problem in the process of context modeling.

Our proposed AA-LSTM can solve this problem by introducing aspect to the process of modeling context to control the flow of information. Information that is important for predicting the given aspect's sentiment polarity can be preserved in the hidden states vectors. In addition, as shown in Equation 6, the alternative input content does not include the aspect information. So the AA-LSTM only utilizes the given aspect to influent the information flow instead of integrating the aspect information into the hidden state vectors.

3.4 Output Gates

The output gate o_t controls the extent of the information flow from the current cell state to the hidden state vector of this time step. Similarly, the aspect-output gate a_o controls the extent of the aspect's influence on the output gate I_t. The difference between our proposed AA-LSTM and the classical LSTM in o_t is the introduction of weighted aspect vector into o_t. And the weight of aspect vector a_o is computed by h_{t-1} and A. Therefore, the degree of how much the aspect information is integrated into o_t is decided by the previous semantic representation and the aspect vector A.

4 Experiment

In this section, we introduce the tasks, the datasets, the evaluation metric, the models for comparison and the implementation details.

4.1 Tasks Definition

We conduct experiments on two subtasks of aspect sentiment analysis: aspect term sentiment analysis (ATSA) or target sentiment analysis (TSA) and aspect category sentiment analysis (ACSA). The former infers sentiment polarities of given target entities contained in the context. The latter infers sentiment polarities of generic aspects such as *'service'* or *'food'* which may or may not be found in the context, and the aspects belong to a predefined set. In this paper, these two kinds of tasks are both considered and they are collectively named as aspect-based sentiment analysis (ABSA).

4.2 Datasets

We experiment on SemEval 2014 [Pontiki *et al.*, 2014] task 4 datasets which consist of *laptop* and *restaurant* reviews and are widely used benchmarks in many previous works [Tang *et al.*, 2016a; Wang *et al.*, 2016; Ma *et al.*, 2017; Chen *et al.*, 2017; Devamanyu *et al.*, 2018; Tay *et al.*, 2018; Wang *et al.*, 2018]. We remove the reviews having no aspect or the aspects with sentiment polarity of "conflict". The dataset we used consists of reviews with at least one aspect labeled with sentiment polarities of *positive, neutral* and *negative*. For ATSA, we adopt Laptop and Restaurant datasets; And for ACSA, we adopt the Restaurant dataset. 20% of the training data is used as the development set. Full statistics of SemEval 2014 task 4 datasets are given in Table 1.

4.3 Evaluation Metric

Since the two tasks are both multi-class classification tasks, we adopt F1-Macro as our evaluation measure. And there are some other methods that use strict accuracy (Acc) [Wang *et al.*, 2016; Ma *et al.*, 2017; Chen *et al.*, 2017; Devamanyu *et al.*, 2018] for evaluation, which measures the percentage of correctly predicted samples in all samples. Therefore, we

Task	Dataset	Pos	Neg	Neu
ATSA	Restaurant Train	2164	807	637
	Restaurant Test	728	196	196
	Laptop Train	994	870	464
	Laptop Test	341	128	169
ACSA	Restaurant Train	2179	839	500
	Restaurant Test	657	222	94

Table 1: Statistic of all datasets

use these two metrics (F1-Macro and Acc) to evaluate the models' performances.

Generally, higher Acc can verify the effectiveness of the system though it biases towards the majority class, and F1-Macro provides more indicative information because the task is a multi-class problem.

4.4 Models for Comparison

In order to verify the advantages of our proposed AA-LSTM compared to classic LSTM, we choose some representative LSTM-based models to replace their original LSTM with our proposed AA-LSTM.

In Introduction and Related Work sections, we have divide recent well-performing methods into three categories according to their processes of modeling context. In order to prove the generalization ability of our model, we select a representative model from each of these categories for experiments. We choose ATAE-LSTM, IAN, and RAM as the representatives of the three categories of models because their architectures are novel and they are taken as comparative methods in many works. We also compare our model with the baseline LSTM model. We introduce them in detail as follows:

LSTM. This is the baseline that ignores targets and only models contexts using one LSTM network. The last hidden state is regarded as the final sentiment representation.

ATAE-LSTM. It concatenates the aspect embedding to the word embeddings of context words and uses aspect embedding to produce the attention vector. For ATSA, we take the average of the embeddings of the target words as the aspect embedding which is concatenated to the word embeddings of the context words.

IAN. It models context and target separately and selects important information from them via two interactive attention mechanisms. The target and context can have impacts on the generation of their representations interactively and their representations are concatenated as the final aspect-specific sentiment representations. For the ACSA task, we omit the modeling of the target and use the aspect embedding to produce the attention vector of context words.

RAM. It utilizes relative location to assign weights to original context hidden state vectors and then learns the attention vector in a recurrent attention mechanism consisting of GRU cell. It can only be applied to ATSA. For the consistent of comparison, we replaced the deep bidirectional LSTM in the original RAM with a unidirectional single-layer LSTM.

We also choose two state-of-the-art methods that are Memory Networks-based and LSTM-based respectively:

Target-sensitive Memory Network. [Wang *et al.*, 2018] construct six target-sensitive memory networks (TMNs) which have their own characteristics to resolve target sensitivity and got some improvement. We choose the NP (hops) and JCI (hops) that perform best on Laptop and Restaurant, respectively.

Inter-Aspect Dependencies LSTM. [Devamanyu *et al.*, 2018] model aspect-based sentential representations as a sequence to capture the inter-aspect dependencies.

We don't reimplement the above two models and the results are retrieved from their original papers.

4.5 Implementation Details

We implement the models in Tensorflow. We initialize all word embeddings by Glove [Jeffrey *et al.*, 2014] and out-of-vocabulary words by sampling from the uniform distribution $U(-0.1, 0.1)$. Initial values of all weight matrices are sampled from uniform distribution $U(-0.1, 0.1)$ and initial values of all biases are zeros. All embedding dimensions are set to 300 and the batch size is set as 16. We minimize the loss function to train our models using Adam optimizer [Diederik and Jimmy, 2014] with the learning rate set as 0.001. To avoid over fitting, we adopt the dropout strategy with $p = 0.5$ and the coefficient of *L2* normalization in the loss function is set to 0.01. All models use softmax classifier.

For ACSA, we initialize all aspect embeddings by sampling from the uniform distribution $U(-0.1, 0.1)$. As for the input aspect vector (A) of our proposed AA-LSTM which is replaced with the classic LSTM in the above models, we set it as follows:

Aspect Term Sentiment Analysis. We use the average of word embeddings of the target words as A except for IAN. For IAN, we use the average of the hidden states vectors of target words as A.

Aspect Category Sentiment Analysis. For all models, we use the aspect embedding as A.

We implement all models under the same experiment settings to make sure the improvements based on the original models come from the replacement of classic LSTM with our proposed AA-LSTM.

5 Results and Analysis

Our experimental results are illustrated in Table 2. We can observe that our proposed AA-LSTM and its substitution in other models has an overall advantage over classic LSTMs on their corresponding original models. It especially achieves higher F1-Macro which can better illustrate the overall performances of the models in multiple classes as the classes are unbalanced. On the ATSA task, except for the F1-Macro score on Restaurant, the performances of our variants overpass the performances of the representative state-of-the-art models. In the implementation of the experiment, the only difference between original models and their variants is the substitution of classic LSTM. As we replace the original LSTM with our AA-LSTM, the performance improvement can demonstrate the pure effectiveness of our AA-LSTM.

Task and Dataset	ATSA				ACSA	
	Laptop		Restaurant		Restaurant	
Model	F1-Macro	Acc	F1-Macro	Acc	F1-Macro	Acc
LSTM	59.77	65.99	61.04	75.00	70.07	81.71
ATAE-LSTM	61.28	66.93	64.47	77.41	70.15	82.12
IAN	64.54	70.53	65.67	78.48	70.81	83.25
RAM	67.05	71.32	65.84	78.57	-	-
IAD-LSTM	-	72.5	-	79.0	-	-
JCI (hops)	67.2	71.8	**68.8**	78.8	-	-
NP (hops)	67.8	72.4	66.0	75.7	-	-
AA-LSTM	61.45	66.93	66.24	78.21	**75.00**	83.45
ATAE-LSTM (AA)	62.10	69.28	66.46	78.21	74.51	83.97
IAN (AA)	65.62	71.94	68.71	**79.29**	74.43	**84.69**
RAM (AA)	**68.47**	**73.20**	68.15	78.13	-	-

Table 2: Comparisons of all models on three datasets. Last four models are our proposed AA-LSTM models, and the last three models with suffix "(AA)" is the variants in which the original classic LSTM is replaced with our proposed AA-LSTM. The results of IAD-LSTM, JCI (hops) and NP (hops) are retrieved from the original papers. Best scores are marked in **bold**.

Compared with LSTM, AA-LSTM's improvements on macro are up to 7% and 6% on Restaurant for ATSA and ACSA respectively. Like LSTM, AA-LSTM also directly uses the last hidden state vector as the final sentiment representation sent to the classifier. But because the aspect is introduced into the process of modeling context, the semantics of the last hidden state vector of AA-LSTM is aspect-specific. In fact, not only in the last hidden layer, but also in all hidden states vectors, the information which is important for determining the emotional polarity of the aspect is kept, and other useless information is filtered, which makes the context modeling result much better than LSTM.As the classifiers are the same, the reason AA-LSTM performs better than LSTM is that the final sentiment representation of AA-LSTM is more effective.

AA-LSTM's performance even surpassed ATAE-LSTM and exceed all original models on F1-macro for ACSA. It is worth mentioning that all baselines utilizes the attention mechanism and ATAE-LSTM also models the context and aspect together via concatenating aspect to every word embeddings of context words. In contrast, AA-LSTM only models the context without any other processing. This proves that AA-LSTM's result of modeling context is aspect-specific and effective. This is because the aspect information is used in the modeling process to control the flow of information, retain and filter information, who performs as earlier attention.

ATAE-LSTM (AA)'s performance exceeds ATAE-LSTM and AA-LSTM. This shows that AA-LSTM can be compatible with other components of ATAE-LSTM, improving the whole model's performance. ATAE-LSTM represents a category of models that combine the context and aspect together via concatenating the aspect vector to context word embeddings. So the experimental results verify that although the input embeddings contain aspect information, it doesn't conflict with the aspect information introduced in AA-LSTM.

IAN represents a category of models which encode the context alone and utilize the aspect to compute contexts' attention vector in the attention mechanism. IAN-LSTM (AA)'s overall performance exceeds IAN and AA-LSTM. This proves that the hidden states vectors generated by AA-LSTM can collaborate with the attention mechanism to achieve better performance.

RAM utilizes the relative location vector to assign weights to original context word hidden state vectors, and calculates the attention vector via a recurrent attention mechanism which is more complex than other baseline models. It is worth noting that compared with RAM, RAM (AA) has more improvement than other original models and their variants. This is because the advantage of AA-LSTM is amplified in RAM. In RAM (AA), while the tokens closer to the target are assigned larger weights, AA-LSTM keeps more important information about the target in the tokens closer to the target: adjectives, modifying phrases, clauses, etc. In addition, the context hidden states vectors generated by AA-LSTM and the recurrent mechanism work together to produce more effective final sentiment representation.

6 Conclusion

In this paper, we argue that aspect-related information may be discarded and aspect-irrelevant information may be retained in classic LSTM cells. To address this problem, we propose a novel LSTM variant termed as Aspect-Aware LSTM. Due to the introduction of the aspect into the process of modeling context, our proposed Aspect-aware LSTM can select important information about the given target and filter out the useless information via information flow control. Aspect-Aware LSTM can not only generate more effective contextual vectors than classic LSTM, but also be compatible with other modules.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments. This work is supported by National Key Research and Development Program of China (Grant No. 2016YFB1000902), National Natural Science Foundation of China (NSFC, Grant Nos. 61866038 and 61751201), and Research Foundation of Beijing Municipal Science and Technology Commissions (No. Z181100008918002).

References

[Bahdanau *et al.*, 2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *Computer Science*, 2014.

[Chen *et al.*, 2017] Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. Recurrent attention network on memory for aspect sentiment analysis. In *Empirical Methods in Natural Language Processing*, pages 452–461, 2017.

[Devamanyu *et al.*, 2018] Hazarika Devamanyu, Poria Soujanya, Vij Prateek, Krishnamurthy Gangeshwar, Cambria Erik, and Zimmermann Roger. Modeling inter-aspect dependencies for aspect-based sentiment analysis. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 266–270, 2018.

[Diederik and Jimmy, 2014] Kingma Diederik and Ba Jimmy. Adam: A method for stochastic optimization. *CoRR abs/1412.6980*, 2014.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Jeffrey *et al.*, 2014] Pennington Jeffrey, Socher Richard, and Manning Christopher. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[Kiritchenko *et al.*, 2014] Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherrt, and Saif Mohammad and. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *International Workshop on Semantic Evaluation*, pages 437–442, 2014.

[Liu and Zhang, 2017] Jiangming Liu and Yue Zhang. Attention modeling for targeted sentiment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, page 572577, 2017.

[Ma *et al.*, 2017] Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. Interactive attention networks for aspect-level sentiment classification. In *International Joint Conference on Artificial Intelligence*, pages 4068–4074, 2017.

[Ma *et al.*, 2018] Yukun Ma, Haiyun Peng, and Erik Cambria. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In *AAAI Conference on Artificial Intelligence*, pages 5876–5883, 2018.

[Pontiki *et al.*, 2014] Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Haris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. Semeval-2014 task 4: Aspect based sentiment analysis. In *International Workshop on Semantic Evaluation*, pages 27–35, 2014.

[Tang *et al.*, 2016a] Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. Effective lstms for target-dependent sentiment classification. In *International Conference on Computational Linguistics*, pages 3298–3307, 2016.

[Tang *et al.*, 2016b] Duyu Tang, Bing Qin, and Ting Liu. Aspect level sentiment classification with deep memory network. In *Empirical Methods in Natural Language Processing*, pages 214–224, 2016.

[Tay *et al.*, 2017] Yi Tay, Tuan Luu Anh, and Hui Siu Cheung. Dyadic memory networks for aspect-based sentiment analysis. In *ACM on Conference on Information and Knowledge Management, CIKM*, pages 107–116, 2017.

[Tay *et al.*, 2018] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Learning to attend via word-aspect associative fusion for aspect-based sentiment analysis. In *AAAI Conference on Artificial Intelligence*, pages 5956–5963, 2018.

[Wagner *et al.*, 2014] Joachim Wagner, Piyush Arora, Santiago Cortes, Utsab Barman, Dasha Bogdanova, Jennifer Foster, and Lamia Tounsi. Dcu: Aspect-based polarity classification for semeval task 4. In *International Workshop on Semantic Evaluation*, pages 223–229, 2014.

[Wang *et al.*, 2016] Yequan Wang, Minlie Huang, and Li Zhao. Attention-based lstm for aspect-level sentiment classification. In *Empirical Methods in Natural Language Processing*, pages 606–615, 2016.

[Wang *et al.*, 2018] Shuai Wang, Sahisnu Mazumder, Bing Liu, Mianwei Zhou, and Yi Chang. Target-sensitive memory networks for aspect sentiment classification. In *Annual Meeting of the Association for Computational Linguistics,*, pages 957–967, 2018.

[Xin *et al.*, 2018] Li Xin, Bing Lidong, Lam Wai, and Shi Bei. Transformation networks for target-oriented sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 946–956, 2018.

[Xu *et al.*, 2017] Jiacheng Xu, Xipeng Qiu, Kan Chen, and Xuanjing Huang. Knowledge graph representation with jointly structural and textual encoding. In *International Joint Conference on Artificial Intelligence*, pages 1318–1324, 2017.

[Xue and Li, 2018] Wei Xue and Tao Li. Aspect based sentiment analysis with gated convolutional networks. In *Annual Meeting of the Association for Computational Linguistics,*, pages 2514–2523, 2018.

[Yang and Mitchell, 2017] Bishan Yang and Tom M. Mitchell. Leveraging knowledge bases in lstms for improving machine reading. In *Annual Meeting of the Association for Computational Linguistics*, pages 1436–1446, 2017.

[Yang *et al.*, 2017] Min Yang, Wenting Tu, Jingxuan Wang, Fei Xu, and Xiaojun Chen. Attention based lstm for target dependent sentiment classification. In *AAAI Conference on Artificial Intelligence*, page 50135014, 2017.

[Zhen *et al.*, 2017] Xu Zhen, Liu Bingquan, Wang Baoxun, Sun Chengjie, and Wang Xiaolong. Incorporating loose-structured knowledge into conversation modeling via recall-gate lstm. In *International Joint Conference on Neural Networks*, pages 3506–3513, 2017.

The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining

Research Track Paper KDD '19, August 4–8, 2019, Anchorage, AK, USA

Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems

Hongwei Wang
Stanford University
hongweiw@cs.stanford.edu

Fuzheng Zhang
Meituan-Dianping Group
zhangfuzheng@meituan.com

Mengdi Zhang
Meituan-Dianping Group
zhangmengdi02@meituan.com

Jure Leskovec
Stanford University
jure@cs.stanford.edu

Miao Zhao, Wenjie Li
Hong Kong Polytechnic University
{csmiaozhao,cswjli}@comp.polyu.edu.hk

Zhongyuan Wang
Meituan-Dianping Group
wangzhongyuan02@meituan.com

ABSTRACT

Knowledge graphs capture structured information and relations between a set of entities or items. As such knowledge graphs represent an attractive source of information that could help improve recommender systems. However, existing approaches in this domain rely on manual feature engineering and do not allow for an end-to-end training. Here we propose *Knowledge-aware Graph Neural Networks with Label Smoothness regularization* (KGNN-LS) to provide better recommendations. Conceptually, our approach computes user-specific item embeddings by first applying a trainable function that identifies important knowledge graph relationships for a given user. This way we transform the knowledge graph into a user-specific weighted graph and then apply a graph neural network to compute personalized item embeddings. To provide better inductive bias, we rely on *label smoothness* assumption, which posits that adjacent items in the knowledge graph are likely to have similar user relevance labels/scores. Label smoothness provides regularization over the edge weights and we prove that it is equivalent to a label propagation scheme on a graph. We also develop an efficient implementation that shows strong scalability with respect to the knowledge graph size. Experiments on four datasets show that our method outperforms state of the art baselines. KGNN-LS also achieves strong performance in cold-start scenarios where user-item interactions are sparse.

KEYWORDS

Knowledge-aware recommendation; graph neural networks; label propagation

ACM Reference Format:
Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3292500.3330836

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330836

1 INTRODUCTION

Recommender systems are widely used in Internet applications to meet user's personalized interests and alleviate the information overload [4, 29, 32]. Traditional recommender systems that are based on collaborative filtering [13, 22] usually suffer from the cold-start problem and have trouble recommending brand new items that have not yet been heavily explored by the users. The sparsity issue can be addressed by introducing additional sources of information such as user/item profiles [23] or social networks [22].

Knowledge graphs (KGs) capture structured information and relations between a set of entities [8, 9, 18, 24–28, 33, 34, 36]. KGs are heterogeneous graphs in which nodes correspond to *entities* (e.g., items or products, as well as their properties and characteristics) and edges correspond to *relations*. KGs provide connectivity information between items via different types of relations and thus capture semantic relatedness between the items.

The core challenge in utilizing KGs in recommender systems is to learn how to capture *user-specific* item-item relatedness captured by the KG. Existing KG-aware recommender systems can be classified into path-based methods [8, 33, 36], embedding-based methods [9, 26, 27, 34], and hybrid methods [18, 24, 28]. However, these approaches rely on manual feature engineering, are unable to perform end-to-end training, and have poor scalability. Graph Neural Networks (GNNs), which aggregate node feature information from node's local network neighborhood using neural networks, represent a promising advancement in graph-based representation learning [3, 5–7, 11, 15]. Recently, several works developed GNNs architecture for recommender systems [14, 19, 28, 31, 32], but these approaches are mostly designed for *homogeneous* bipartite user-item interaction graphs or user-/item-similarity graphs. It remains an open question how to extend GNNs architecture to *heterogeneous* knowledge graphs.

In this paper, we develop *Knowledge-aware Graph Neural Networks with Label Smoothness regularization* (KGNN-LS) that extends GNNs architecture to knowledge graphs to simultaneously capture semantic relationships between the items as well as personalized user preferences and interests. To account for the relational heterogeneity in KGs, similar to [28], we use a trainable and personalized relation scoring function that transforms the KG into a user-specific weighted graph, which characterizes both the semantic information of the KG as well as user's personalized interests. For example, in the movie recommendation setting the relation scoring function could learn that a given user really cares about "director" relation between movies and persons, while somebody else may care more

about the "lead actor" relation. Using this personalized weighted graph, we then apply a graph neural network that for every item node computes its embedding by aggregating node feature information over the local network neighborhood of the item node. This way the embedding of each item captures it's local KG structure in a user-personalized way.

A significant difference between our approach and traditional GNNs is that the edge weights in the graph are not given as input. We set them using user-specific relation scoring function that is trained in a supervised fashion. However, the added flexibility of edge weights makes the learning process prone to overfitting, since the only source of supervised signal for the relation scoring function is coming from user-item interactions (which are sparse in general). To remedy this problem, we develop a technique for regularization of edge weights during the learning process, which leads to better generalization. We develop an approach based on *label smoothness* [35, 38], which assumes that adjacent entities in the KG are likely to have similar user relevancy labels/scores. In our context this assumption means that users tend to have similar preferences to items that are nearby in the KG. We prove that label smoothness regularization is equivalent to *label propagation* and we design a *leave-one-out* loss function for label propagation to provide extra supervised signal for learning the edge scoring function. We show that the knowledge-aware graph neural networks and label smoothness regularization can be unified under the same framework, where label smoothness can be seen as a natural choice of regularization on knowledge-aware graph neural networks.

We apply the proposed method to four real-world datasets of movie, book, music, and restaurant recommendations, in which the first three datasets are public datasets and the last is from Meituan-Dianping Group. Experiments show that our method achieves significant gains over state-of-the-art methods in recommendation accuracy. We also show that our method maintains strong recommendation performance in the cold-start scenarios where user-item interactions are sparse.

2 RELATED WORK

2.1 Graph Neural Networks

Graph Neural Networks (or Graph Convolutional Neural Networks, GCNs) aim to generalize convolutional neural networks to non-Euclidean domains (such as graphs) for robust feature learning. Bruna et al. [3] define the convolution in Fourier domain and calculate the eigendecomposition of the graph Laplacian, Defferrard et al. [5] approximate the convolutional filters by Chebyshev expansion of the graph Laplacian, and Kipf et al. [11] propose a convolutional architecture via a first-order approximation. In contrast to these *spectral* GCNs, *non-spectral* GCNs operate on the graph directly and apply "convolution" (i.e., weighted average) to local neighbors of a node [6, 7, 15].

Recently, researchers also deployed GCNs in recommender systems: PinSage [32] applies GCNs to the pin-board bipartite graph in Pinterest. Monti et al. [14] and Berg et al. [19] model recommender systems as matrix completion and design GCNs for representation learning on user-item bipartite graphs. Wu et al. [31] use GCNs on user/item structure graphs to learn user/item representations. The difference between these works and ours is that they are all designed for homogeneous bipartite graphs or user/item-similarity graphs where GCNs can be used directly, while here we investigate GCNs for heterogeneous KGs. Wang et al. [28] use GCNs in KGs for recommendation, but simply applying GCNs to KGs without proper regularization is prone to overfitting and leads to performance degradation as we will show later. Schlichtkrull et al. also propose using GCNs to model KGs [17], but not for the purpose of recommendations.

2.2 Semi-supervised Learning on Graphs

The goal of graph-based semi-supervised learning is to correctly label all nodes in a graph given that only a few nodes are labeled. Prior work often makes assumptions on the distribution of labels over the graph, and one common assumption is smooth variation of labels of nodes across the graph. Based on different settings of edge weights in the input graph, these methods are classified as: (1) Edge weights are assumed to be given as input and therefore fixed [1, 37, 38]; (2) Edge weights are parameterized and therefore learnable [10, 21, 35]. Inspired by these methods, we design a module of label smoothness regularization in our proposed model. The major distinction of our work is that the label smoothness constraint is not used for semi-supervised learning on graphs, but serves as regularization to assist the learning of edge weights and achieves better generalization for recommender systems.

2.3 Recommendations with Knowledge Graphs

In general, existing KG-aware recommender systems can be classified into three categories: (1) *Embedding-based methods* [9, 26, 27, 34] pre-process a KG with *knowledge graph embedding* (KGE) [30] algorithms, then incorporate learned entity embeddings into recommendation. Embedding-based methods are highly flexible in utilizing KGs to assist recommender systems, but the KGE algorithms focus more on modeling rigorous semantic relatedness (e.g., TransE [2] assumes $head + relation = tail$), which are more suitable for graph applications such as link prediction rather than recommendations. In addition, embedding-based methods usually lack an end-to-end way of training. (2) *Path-based methods* [8, 33, 36] explore various patterns of connections among items in a KG (a.k.a meta-path or meta-graph) to provide additional guidance for recommendations. Path-based methods make use of KGs in a more intuitive way, but they rely heavily on manually designed meta-paths/meta-graphs, which are hard to tune in practice. (3) *Hybrid methods* [18, 24, 28] combine the above two categories and learn user/item embeddings by exploiting the structure of KGs. Our proposed model can be seen as an instance of hybrid methods.

3 PROBLEM FORMULATION

We begin by describing the KG-aware recommendations problem and introducing notation. In a typical recommendation scenario, we have a set of users $\mathcal{U}$ and a set of items $\mathcal{V}$. The user-item interaction matrix $\mathbf{Y}$ is defined according to users' implicit feedback, where $y_{uv} = 1$ indicates that user u has engaged with item v, such as clicking, watching, or purchasing. We also have a knowledge graph $\mathcal{G} = \{(h, r, t)\}$ available, in which $h \in \mathcal{E}$, $r \in \mathcal{R}$, and $t \in \mathcal{E}$ denote the head, relation, and tail of a knowledge triple, $\mathcal{E}$ and $\mathcal{R}$ are the set of entities and relations in the knowledge graph, respectively. For

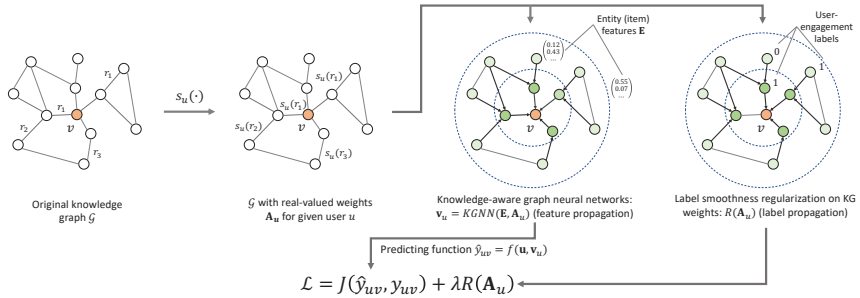


Figure 1: Overview of our proposed KGNN-LS model. The original KG is first transformed into a user-specific weighted graph, on which we then perform feature propagation using a graph neural network with the label smoothness regularization. The two modules constitute the complete loss function $\mathcal{L}$.

example, the triple (*The Silence of the Lambs*, *film.film.star*, *Anthony Hopkins*) states the fact that Anthony Hopkins is the leading actor in film "The Silence of the Lambs". In many recommendation scenarios, an item $v \in \mathcal{V}$ corresponds to an entity $e \in \mathcal{E}$ (e.g., item "The Silence of the Lambs" in MovieLens also appears in the knowledge graph as an entity). The set of entities $\mathcal{E}$ is composed from items $\mathcal{V}$ ($\mathcal{V} \subseteq \mathcal{E}$) as well as non-items $\mathcal{E} \backslash \mathcal{V}$ (e.g. nodes corresponding to item/product properties). Given user-item interaction matrix $\mathbf{Y}$ and knowledge graph $\mathcal{G}$, our task is to predict whether user u has potential interest in item v with which he/she has not engaged before. Specifically, we aim to learn a prediction function $\hat{y}_{uv} = \mathcal{F}(u, v|\Theta, \mathbf{Y}, \mathcal{G})$, where $\hat{y}_{uv}$ denotes the probability that user u will engage with item v, and Θ are model parameters of function $\mathcal{F}$.

We list the key symbols used in this paper in Table 1.

Symbol	Meaning
$\mathcal{U} = \{u_1, \cdots\}$	Set of users
$\mathcal{V} = \{v_1, \cdots\}$	Set of items
$\mathbf{Y}$	User-item interaction matrix
$\mathcal{G} = (\mathcal{E}, \mathcal{R})$	Knowledge graph
$\mathcal{E} = \{e_1, \cdots\}$	Set of entities
$\mathcal{R} = \{r_1, \cdots\}$	Set of relations
$\mathcal{E} \backslash \mathcal{V}$	Set of non-item entities
$s_u(r)$	User-specific relation scoring function
$\mathbf{A}_u$	Adjacency matrix of $\mathcal{G}$ w.r.t. user u
$\mathbf{D}_u$	Diagonal degree matrix of $\mathbf{A}_u$
$\mathbf{E}$	Raw entity feature
$\mathbf{H}^{(l)},\ l = 0, ..., L-1$	Entity representation in the l-th layer
$\mathbf{W}^{(l)},\ l = 0, ..., L-1$	Transformation matrix in the l-th layer
$l_u(e),\ e \in \mathcal{E}$	Item relevancy labeling function
$l_u^*(e),\ e \in \mathcal{E}$	Minimum-energy labeling function
$\hat{l}_u(v),\ v \in \mathcal{V}$	Predicted relevancy label for item v
$R(\mathbf{A}_u)$	Label smoothness regularization on $\mathbf{A}_u$

Table 1: List of key symbols.

4 OUR APPROACH

In this section, we first introduce knowledge-aware graph neural networks and label smoothness regularization, respectively, then we present the unified model.

4.1 Preliminaries: Knowledge-aware Graph Neural Networks

The first step of our approach is to transform a heterogeneous KG into a user-personalized weighted graph that characterizes user's preferences. To this end, similar to [28], we use a user-specific *relation scoring function* $s_u(r)$ that provides the importance of relation r for user u: $s_u(r) = g(\mathbf{u}, \mathbf{r})$, where $\mathbf{u}$ and $\mathbf{r}$ are feature vectors of user u and relation type r, respectively, and g is a differentiable function such as inner product. Intuitively, $s_u(r)$ characterizes the importance of relation r to user u. For example, a user may be more interested in *directors* of movies, but another user may care more about the *lead actors* of movies.

Given user-specific relation scoring function $s_u(\cdot)$ of user u, knowledge graph $\mathcal{G}$ can therefore be transformed into a user-specific adjacency matrix $\mathbf{A}_u \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{E}|}$, in which the (i, j)-entry $A_u^{ij} = s_u(r_{e_i,e_j})$, and r_{e_i,e_j} is the relation between entities e_i and e_j in $\mathcal{G}$.[1] $A_u^{ij} = 0$ if there is no relation between e_i and e_j. See the left two subfigures in Figure 1 for illustration. We also denote the raw feature matrix of entities as $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d_0}$, where d_0 is the dimension of raw entity features. Then we use multiple feed forward layers[2] to update the entity representation matrix by aggregating representations of neighboring entities. Specifically, the layer-wise forward propagation can be expressed as

$$\mathbf{H}_{l+1} = \sigma\left(\mathbf{D}_u^{-1/2}\mathbf{A}_u\mathbf{D}_u^{-1/2}\mathbf{H}_l\mathbf{W}_l\right),\ l = 0, 1, \cdots, L-1. \quad (1)$$

[1] In this work we treat $\mathcal{G}$ an undirected graph, so $\mathbf{A}_u$ is a symmetric matrix. If both triples (h, r_1, t) and (t, r_2, h) exist, we only consider one of r_1 and r_2. This is due to the fact that: (1) r_1 and r_2 are the inverse of each other and semantically related; (2) Treating $\mathbf{A}_u$ symmetric will greatly increase the matrix density.

[2] There are several candidate designs for the architecture of our model, e.g., GCN [11] or GraphSAGE [7]. Here we use GCN [11] as our base model.

In Eq. (1), $\mathbf{H}_l$ is the matrix of hidden representations of entities in layer l, and $\mathbf{H}_0 = \mathbf{E}$. $\mathbf{A}_u$ is to aggregate representation vectors of neighboring entities. In this paper, we set $\mathbf{A}_u \leftarrow \mathbf{A}_u + \mathbf{I}$, i.e., adding self-connection to each entity, to ensure that old representation vector of the entity itself is taken into consideration when updating entity representations. $\mathbf{D}_u$ is a diagonal degree matrix with entries $D_u^{ii} = \sum_j A_u^{ij}$, therefore, $\mathbf{D}_u^{-1/2}$ is used to normalize $\mathbf{A}_u$ and keep the entity representation matrix $\mathbf{H}_l$ stable. $\mathbf{W}_l \in \mathbb{R}^{d_l \times d_{l+1}}$ is the layer-specific trainable weight matrix, σ is a non-linear activation function, and L is the number of layers.

A single GNN layer computes the representation of an entity via a transformed mixture of itself and its immediate neighbors in the KG. We can therefore naturally extend the model to multiple layers to explore users' potential interests in a broader and deeper way. The final output is $\mathbf{H}_L \in \mathbb{R}^{|\mathcal{E}| \times d_L}$, which is the entity representations that mix the initial features of themselves and their neighbors up to L hops away. Finally, the predicted engagement probability of user u with item v is calculated by $\hat{y}_{uv} = f(\mathbf{u}, \mathbf{v}_u)$, where $\mathbf{v}_u$ (i.e., the v-th row of $\mathbf{H}_L$) is the final representation vector of item v, and f is a differentiable prediction function, for example, inner product or a multilayer perceptron. Note that $\mathbf{v}_u$ is user-specific since the adjacency matrix $\mathbf{A}_u$ is user-specific. Furthermore, note that the system is end-to-end trainable where the gradients flow from $f(\cdot)$ via GNN (parameter matrix $\mathbf{W}$) to $g(\cdot)$ and eventually to representations of users u and items v.

4.2 Label Smoothness Regularization

It is worth noticing a significant difference between our model and GNNs: In traditional GNNs, edge weights of the input graph are fixed; but in our model, edge weights $\mathbf{D}_u^{-1/2}\mathbf{A}_u\mathbf{D}_u^{-1/2}$ in Eq. (1) are learnable (including possible parameters of function g and feature vectors of users and relations) and also requires supervised training like $\mathbf{W}$. Though enhancing the fitting ability of the model, this will inevitably make the optimization process prone to overfitting, since the only source of supervised signal is from user-item interactions outside GNN layers. Moreover, edge weights do play an essential role in representation learning on graphs, as highlighted by a large amount of prior works [10, 20, 21, 35, 38]. Therefore, more regularization on edge weights is needed to assist the learning of entity representations and to help generalize to unobserved interactions more efficiently.

Let's see how an ideal set of edge weights should be like. Consider a real-valued label function $l_u : \mathcal{E} \to \mathbb{R}$ on $\mathcal{G}$, which is constrained to take a specific value $l_u(v) = y_{uv}$ at node $v \in \mathcal{V} \subseteq \mathcal{E}$. In our context, $l_u(v) = 1$ if user u finds the item v relevant and has engaged with it, otherwise $l_u(v) = 0$. Intuitively, we hope that adjacent entities in the KG are likely to have similar relevancy labels, which is known as *label smoothness assumption*. This motivates our choice of energy function E:

$$E(l_u, \mathbf{A}_u) = \frac{1}{2} \sum_{e_i \in \mathcal{E}, e_j \in \mathcal{E}} A_u^{ij} \left(l_u(e_i) - l_u(e_j)\right)^2. \tag{2}$$

We show that the minimum-energy label function is *harmonic* by the following theorem:

Theorem 1. *The minimum-energy label function*

$$l_u^* = \operatorname*{arg\,min}_{l_u : l_u(v) = y_{uv}, \forall v \in \mathcal{V}} E(l_u, \mathbf{A}_u) \tag{3}$$

w.r.t. Eq. (2) is harmonic, i.e., l_u^* satisfies

$$l_u^*(e_i) = \frac{1}{D_u^{ii}} \sum_{e_j \in \mathcal{E}} A_u^{ij}\, l_u^*(e_j),\ \forall e_i \in \mathcal{E} \backslash \mathcal{V}. \tag{4}$$

Proof. Taking the derivative of the following equation

$$E(l_u, \mathbf{A}_u) = \frac{1}{2} \sum_{i,j} A_u^{ij} \left(l_u(e_i) - l_u(e_j)\right)^2$$

with respect to $l_u(e_i)$ where $e_i \in \mathcal{E} \backslash \mathcal{V}$, we have

$$\frac{\partial E(l_u, \mathbf{A}_u)}{\partial l_u(e_i)} = \sum_j A_u^{ij} \left(l_u(e_i) - l_u(e_j)\right).$$

The minimum-energy label function l_u^* should satisfy that

$$\left.\frac{\partial E(l_u, \mathbf{A}_u)}{\partial l_u(e_i)}\right|_{l_u = l_u^*} = 0.$$

Therefore, we have

$$l_u^*(e_i) = \frac{1}{\sum_j A_u^{ij}} \sum_j A_u^{ij}\, l_u^*(e_j) = \frac{1}{D_u^{ii}} \sum_j A_u^{ij}\, l_u^*(e_j),\ \forall e_i \in \mathcal{E} \backslash \mathcal{V}.$$

□

The harmonic property indicates that the value of l_u^* at each non-item entity $e_i \in \mathcal{E} \backslash \mathcal{V}$ is the average of its neighboring entities, which leads to the following label propagation scheme [39]:

Theorem 2. *Repeating the following two steps:*

(1) Propagate labels for all entities: $l_u(\mathcal{E}) \leftarrow \mathbf{D}_u^{-1}\mathbf{A}_u l_u(\mathcal{E})$, where $l_u(\mathcal{E})$ is the vector of labels for all entities;

(2) Reset labels of all items to initial labels: $l_u(\mathcal{V}) \leftarrow \mathbf{Y}[u, \mathcal{V}]^\top$, where $l_u(\mathcal{V})$ is the vector of labels for all items and $\mathbf{Y}[u, \mathcal{V}] = [y_{uv_1}, y_{uv_2}, \cdots]$ are initial labels;

will lead to $l_u \to l_u^*$.

Proof. Let $l_u(\mathcal{E}) = \begin{bmatrix} l_u(\mathcal{V}) \\ l_u(\mathcal{E} \backslash \mathcal{V}) \end{bmatrix}$. Since $l_u(\mathcal{V})$ is fixed on $\mathbf{Y}[u, \mathcal{V}]$, we are only interested in $l_u(\mathcal{E} \backslash \mathcal{V})$. We denote $\mathbf{P} = \mathbf{D}_u^{-1}\mathbf{A}_u$ (the subscript u is omitted from $\mathbf{P}$ for ease of notation), and partition matrix $\mathbf{P}$ into sub-matrices according to the partition of l_u:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{VV} & \mathbf{P}_{VE} \\ \mathbf{P}_{EV} & \mathbf{P}_{EE} \end{bmatrix}.$$

Then the label propagation scheme is equivalent to

$$l_u(\mathcal{E} \backslash \mathcal{V}) \leftarrow \mathbf{P}_{EV}\mathbf{Y}[u, \mathcal{V}]^\top + \mathbf{P}_{EE} l_u(\mathcal{E} \backslash \mathcal{V}). \tag{5}$$

Repeat the above procedure, we have

$$l_u(\mathcal{E} \backslash \mathcal{V}) = \lim_{n \to \infty} (\mathbf{P}_{EE})^n l_u^{(0)}(\mathcal{E} \backslash \mathcal{V}) + \left(\sum_{i=1}^{n} (\mathbf{P}_{EE})^{i-1}\right) \mathbf{P}_{EV}\mathbf{Y}[u, \mathcal{V}]^\top, \tag{6}$$

where $l_u^{(0)}(\mathcal{E} \backslash \mathcal{V})$ is the initial value for $l_u(\mathcal{E} \backslash \mathcal{V})$. Now we show that $lim_{n \to \infty} (\mathbf{P}_{EE})^n l_u^{(0)}(\mathcal{E} \backslash \mathcal{V}) = \mathbf{0}$. Since $\mathbf{P}$ is row-normalized and $\mathbf{P}_{EE}$ is a sub-matrix of $\mathbf{P}$, we have

$$\exists \epsilon < 1,\ \sum_j \mathbf{P}_{EE}[i, j] \le \epsilon,$$

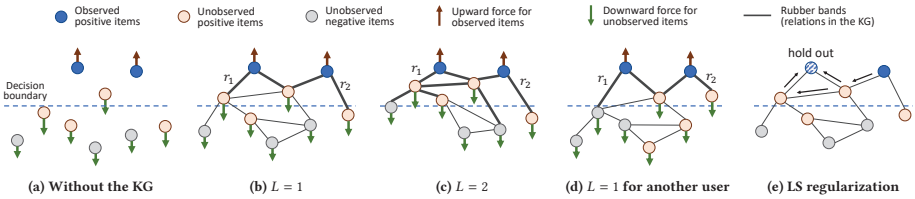


Figure 2: (a) Analogy of a physical equilibrium model for recommender systems; (b)-(d) Illustration of the effect of the KG; (e) Illustration of the effect of label smoothness regularization.

for all possible row index i. Therefore,

$$\begin{aligned}\sum_j (\mathbf{P}_{EE})^n[i,j] &= \sum_j \left((\mathbf{P}_{EE})^{(n-1)}\mathbf{P}_{EE}\right)[i,j] \\ &= \sum_j \sum_k (\mathbf{P}_{EE})^{(n-1)}[i,k]\,\mathbf{P}_{EE}[k,j] \\ &= \sum_k (\mathbf{P}_{EE})^{(n-1)}[i,k] \sum_j \mathbf{P}_{EE}[k,j] \\ &\le \sum_k (\mathbf{P}_{EE})^{(n-1)}[i,k]\,\epsilon \\ &\le \cdots \le \epsilon^n.\end{aligned}$$

As n goes infinity, the row sum of $(\mathbf{P}_{EE})^n$ converges to zero, which implies that $(\mathbf{P}_{EE})^n l_u^{(0)}(\mathcal{E}\backslash\mathcal{V}) \to \mathbf{0}$. It's clear that the choice of initial value $l_u^{(0)}(\mathcal{E}\backslash\mathcal{V})$ does not affect the convergence.

Since $lim_{n\to\infty}(\mathbf{P}_{EE})^n l_u^{(0)}(\mathcal{E}\backslash\mathcal{V}) = \mathbf{0}$, Eq. (6) becomes

$$l_u(\mathcal{E}\backslash\mathcal{V}) = \lim_{n\to\infty}\left(\sum_{i=1}^{n}(\mathbf{P}_{EE})^{i-1}\right)\mathbf{P}_{EV}\mathbf{Y}[u,\mathcal{V}]^\top.$$

Denote

$$\mathbf{T} = \lim_{n\to\infty}\sum_{i=1}^{n}(\mathbf{P}_{EE})^{i-1} = \sum_{i=1}^{\infty}(\mathbf{P}_{EE})^{i-1},$$

and we have

$$\mathbf{T} - \mathbf{T}\mathbf{P}_{EE} = \sum_{i=1}^{\infty}(\mathbf{P}_{EE})^{i-1} - \sum_{i=1}^{\infty}(\mathbf{P}_{EE})^{i} = \mathbf{I}.$$

Therefore, we derive that

$$\mathbf{T} = (I - \mathbf{P}_{EE})^{-1},$$

and

$$l_u(\mathcal{E}\backslash\mathcal{V}) = (I - \mathbf{P}_{EE})^{-1}\mathbf{P}_{EV}\mathbf{Y}[u,\mathcal{V}]^\top.$$

This is the unique fixed point and therefore the unique solution to Eq. (5). Repeating the steps in Theorem 2 leads to

$$l_u(\mathcal{E}) \to l_u^*(\mathcal{E}) = \begin{bmatrix}\mathbf{Y}[u,\mathcal{V}]^\top \\ (I - \mathbf{P}_{EE})^{-1}\mathbf{P}_{EV}\mathbf{Y}[u,\mathcal{V}]^\top\end{bmatrix}.$$

□

Theorem 2 provides a way for reaching the minimum-energy of relevancy label function E. However, l_u^* does not provide any signal for updating the edge weights matrix $\mathbf{A}_u$, since the labeled part of l_u^*, i.e., $l_u^*(\mathcal{V})$, equals their true relevancy labels $\mathbf{Y}[u,\mathcal{V}]$; Moreover, we do not know true relevancy labels for the unlabeled nodes $l_u^*(\mathcal{E}\backslash\mathcal{V})$.

To solve the issue, we propose minimizing the *leave-one-out* loss [35]. Suppose we hold out a single item v and treat it unlabeled. Then we predict its label by using the rest of (labeled) items and (unlabeled) non-item entities. The prediction process is identical to label propagation in Theorem 2, except that the label of item v is hidden and needs to be calculated. This way, the difference between the true relevancy label of v (i.e., y_{uv}) and the predicted label $\hat{l}_u(v)$ serves as a supervised signal for regularizing edge weights:

$$R(\mathbf{A}) = \sum_u R(\mathbf{A}_u) = \sum_u \sum_v J\big(y_{uv}, \hat{l}_u(v)\big), \tag{7}$$

where J is the cross-entropy loss function. Given the regularization in Eq. (7), an ideal edge weight matrix $\mathbf{A}$ should reproduce the true relevancy label of each held-out item while also satisfying the smoothness of relevancy labels.

4.3 The Unified Loss Function

Combining knowledge-aware graph neural networks and LS regularization, we reach the following complete loss function:

$$\min_{\mathbf{W},\mathbf{A}} \mathcal{L} = \min_{\mathbf{W},\mathbf{A}} \sum_{u,v} J(y_{uv}, \hat{y}_{uv}) + \lambda R(\mathbf{A}) + \gamma\|\mathcal{F}\|_2^2, \tag{8}$$

where $\|\mathcal{F}\|_2^2$ is the L2-regularizer, λ and γ are balancing hyper-parameters. In Eq. (8), the first term corresponds to the part of GNN that learns the transformation matrix $\mathbf{W}$ and edge weights $\mathbf{A}$ simultaneously, while the second term $R(\cdot)$ corresponds to the part of label smoothness that can be seen as adding constraint on edge weights $\mathbf{A}$. Therefore, $R(\cdot)$ serves as regularization on $\mathbf{A}$ to assist GNN in learning edge weights.

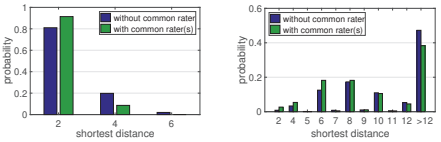
It is also worth noticing that the first term can be seen as *feature propagation* on the KG while the second term $R(\cdot)$ can be seen as *label propagation* on the KG. A recommender for a specific user u is actually a mapping from item features to user-item interaction labels, i.e., $\mathcal{F}_u : \mathbf{E}_v \to y_{uv}$ where $\mathbf{E}_v$ is the feature vector of item v. Therefore, Eq. (8) utilizes the structural information of the KG on both the feature side and the label side of $\mathcal{F}_u$ to capture users' higher-order preferences.

4.4 Discussion

How can the knowledge graph help find users' interests? To intuitively understand the role of the KG, we make an analogy with a

	Movie	Book	Music	Restaurant
# users	138,159	19,676	1,872	2,298,698
# items	16,954	20,003	3,846	1,362
# interactions	13,501,622	172,576	42,346	23,416,418
# entities	102,569	25,787	9,366	28,115
# relations	32	18	60	7
# KG triples	499,474	60,787	15,518	160,519

Table 2: Statistics of the four datasets: MovieLens-20M (movie), Book-Crossing (book), Last.FM (music), and Dianping-Food (restaurant).



(a) MovieLens-20M **(b) Last.FM**

Figure 3: Probability distribution of the shortest path distance between two randomly sampled items in the KG under the circumstance that (1) they have no common user in the dataset; (2) they have common user(s) in the dataset.

physical equilibrium model as shown in Figure 2. Each entity/item is seen as a particle, while the supervised positive user-relevancy signal acts as the force pulling the observed positive items up from the decision boundary and the negative items signal acts as the force pushing the unobserved items down. Without the KG (Figure 2a), these items are only loosely connected with each other through the collaborative filtering effect (which is not drawn here for clarity). In contrast, edges in the KG serve as the rubber bands that impose explicit constraints on connected entities. When number of layers is $L = 1$ (Figure 2b), representation of each entity is a mixture of itself and its immediate neighbors, therefore, optimizing on the positive items will simultaneously pull their immediate neighbors up together. The upward force goes deeper in the KG with the increase of L (Figure 2c), which helps explore users' long-distance interests and pull up more positive items. It is also interesting to note that the proximity constraint exerted by the KG is *personalized* since the strength of the rubber band (i.e., $s_u(r)$) is *user-specific* and *relation-specific*: One user may prefer relation r_1 (Figure 2b) while another user (with same observed items but different unobserved items) may prefer relation r_2 (Figure 2d).

Despite the force exerted by edges in the KG, edge weights may be set inappropriately, for example, too small to pull up the unobserved items (i.e., rubber bands are too weak). Next, we show by Figure 2e that how the label smoothness assumption helps regularizing the learning of edge weights. Suppose we hold out the positive sample in the upper left and we intend to reproduce its label by the rest of items. Since the true relevancy label of the held-out sample is 1 and the upper right sample has the largest label value, the LS regularization term $R(\mathbf{A})$ would enforce the edges with arrows to be large so that the label can "flow" from the blue one to the striped one as much as possible. As a result, this will tighten the rubber bands (denoted by arrows) and encourage the model to pull up the two upper pink items to a greater extent.

5 EXPERIMENTS

In this section, we evaluate the proposed KGNN-LS model, and present its performance on four real-world scenarios: movie, book, music, and restaurant recommendations.

5.1 Datasets

We utilize the following four datasets in our experiments for movie, book, music, and restaurant recommendations, respectively, in which the first three are public datasets and the last one is from Meituan-Dianping Group. We use Satori[3], a commercial KG built by Microsoft, to construct sub-KGs for MovieLens-20M, Book-Crossing, and Last.FM datasets. The KG for Dianping-Food dataset is constructed by the internal toolkit of Meituan-Dianping Group. Further details of datasets are provided in Appendix A.

- **MovieLens-20M**[4] is a widely used benchmark dataset in movie recommendations, which consists of approximately 20 million explicit ratings (ranging from 1 to 5) on the MovieLens website. The corresponding KG contains 102,569 entities, 499,474 edges and 32 relation-types.
- **Book-Crossing**[5] contains 1 million ratings (ranging from 0 to 10) of books in the Book-Crossing community. The corresponding KG contains 25,787 entities, 60,787 edges and 18 relation-types.
- **Last.FM**[6] contains musician listening information from a set of 2 thousand users from Last.fm online music system. The corresponding KG contains 9,366 entities, 15,518 edges and 60 relation-types.
- **Dianping-Food** is provided by Dianping.com[7], which contains over 10 million interactions (including clicking, buying, and adding to favorites) between approximately 2 million users and 1 thousand restaurants. The corresponding KG contains 28,115 entities, 160,519 edges and 7 relation-types.

The statistics of the four datasets are shown in Table 2.

5.2 Baselines

We compare the proposed KGNN-LS model with the following baselines for recommender systems, in which the first two baselines are KG-free while the rest are all KG-aware methods. The hyper-parameter setting of KGNN-LS is provided in Appendix B.

- **SVD** [12] is a classic CF-based model using inner product to model user-item interactions. We use the unbiased version (i.e., the predicted engaging probability is modeled as $y_{uv} = \mathbf{u}^\top \mathbf{v}$). The dimension and learning rate for the four datasets are set as: $d = 8$, $\eta = 0.5$ for MovieLens-20M, Book-Crossing; $d = 8$, $\eta = 0.1$ for Last.FM; $d = 32$, $\eta = 0.1$ for Dianping-Food.

[3] https://searchengineland.com/library/bing/bing-satori
[4] https://grouplens.org/datasets/movielens/
[5] http://www2.informatik.uni-freiburg.de/~cziegler/BX/
[6] https://grouplens.org/datasets/hetrec-2011/
[7] https://www.dianping.com/

Model	MovieLens-20M				Book-Crossing				Last.FM				Dianping-Food			
	R@2	*R@10*	*R@50*	*R@100*	*R@2*	*R@10*	*R@50*	*R@100*	*R@2*	*R@10*	*R@50*	*R@100*	*R@2*	*R@10*	*R@50*	*R@100*
SVD	0.036	0.124	0.277	0.401	0.027	0.046	0.077	0.109	0.029	0.098	0.240	0.332	0.039	0.152	0.329	0.451
LibFM	0.039	0.121	0.271	0.388	0.033	0.062	0.092	0.124	0.030	0.103	0.263	0.330	0.043	0.156	0.332	0.448
LibFM + TransE	0.041	0.125	0.280	0.396	0.037	0.064	0.097	0.130	0.032	0.102	0.259	0.326	0.044	0.161	**0.343**	0.455
PER	0.022	0.077	0.160	0.243	0.022	0.041	0.064	0.070	0.014	0.052	0.116	0.176	0.023	0.102	0.256	0.354
CKE	0.034	0.107	0.244	0.322	0.028	0.051	0.079	0.112	0.023	0.070	0.180	0.296	0.034	0.138	0.305	0.437
RippleNet	**0.045**	0.130	0.278	0.447	0.036	0.074	0.107	0.127	0.032	0.101	0.242	0.336	0.040	0.155	0.328	0.440
KGNN-LS	0.043	**0.155**	**0.321**	**0.458**	**0.045**	**0.082**	**0.117**	**0.149**	**0.044**	**0.122**	**0.277**	**0.370**	**0.047**	**0.170**	0.340	**0.487**

Table 3: The results of *Recall@K* in top-K recommendation.

Model	Movie	Book	Music	Restaurant
SVD	0.963	0.672	0.769	0.838
LibFM	0.959	0.691	0.778	0.837
LibFM + TransE	0.966	0.698	0.777	0.839
PER	0.832	0.617	0.633	0.746
CKE	0.924	0.677	0.744	0.802
RippleNet	0.960	0.727	0.770	0.833
KGNN-LS	**0.979**	**0.744**	**0.803**	**0.850**

Table 4: The results of *AUC* in CTR prediction.

- **LibFM** [16] is a widely used feature-based factorization model for CTR prediction. We concatenate user ID and item ID as input for LibFM. The dimension is set as $\{1, 1, 8\}$ and the number of training epochs is 50 for all datasets.
- **LibFM + TransE** extends LibFM by attaching an entity representation learned by TransE [2] to each user-item pair. The dimension of TransE is 32 for all datasets.
- **PER** [33] is a representative of path-based methods, which treats the KG as heterogeneous information networks and extracts meta-path based features to represent the connectivity between users and items. We use manually designed "user-item-attribute-item" as meta-paths, i.e., "user-movie-director-movie", "user-movie-genre-movie", and "user-movie-star-movie" for MovieLens-20M; "user-book-author-book" and "user-book-genre-book" for Book-Crossing, "user-musician-date_of_birth-musician" (date of birth is discretized), "user-musician-country-musician", and "user-musician-genre-musician" for Last.FM; "user-restaurant-dish-restaurant", "user-restaurant-business_area-restaurant", "user-restaurant-tag-restaurant" for Dianping-Food. The settings of dimension and learning rate are the same as SVD.
- **CKE** [34] is a representative of embedding-based methods, which combines CF with structural, textual, and visual knowledge in a unified framework. We implement CKE as CF plus a structural knowledge module in this paper. The dimension of embedding for the four datasets are 64, 128, 64, 64. The training weight for KG part is 0.1 for all datasets. The learning rate are the same as in SVD.
- **RippleNet** [24] is a representative of hybrid methods, which is a memory-network-like approach that propagates users' preferences on the KG for recommendation. For RippleNet, $d = 8$, $H = 2$, $\lambda_1 = 10^{-6}$, $\lambda_2 = 0.01$, $\eta = 0.01$ for MovieLens-20M; $d = 16$, $H = 3$, $\lambda_1 = 10^{-5}$, $\lambda_2 = 0.02$, $\eta = 0.005$ for Last.FM; $d = 32$, $H = 2$, $\lambda_1 = 10^{-7}$, $\lambda_2 = 0.02$, $\eta = 0.01$ for Dianping-Food.

5.3 Validating the Connection between $\mathcal{G}$ and Y

To validate the connection between the knowledge graph $\mathcal{G}$ and user-item interaction Y, we conduct an empirical study where we investigate the correlation between the shortest path distance of two randomly sampled items in the KG and whether they have common user(s) in the dataset, that is there exist user(s) that interacted with both items. For MovieLens-20M and Last.FM, we randomly sample ten thousand item pairs that have no common users and have at least one common user, respectively, then count the distribution of their shortest path distances in the KG. The results are presented in Figure 3, which clearly show that *if two items have common user(s) in the dataset, they are likely to be more close in the KG*. For example, if two movies have common user(s) in MovieLens-20M, there is a probability of 0.92 that they will be within 2 hops in the KG, while the probability is 0.80 if they have no common user. This finding empirically demonstrates that exploiting the proximity structure of the KG can assist making recommendations. This also justifies our motivation to use label smoothness regularization to help learn entity representations.

5.4 Results

5.4.1 Comparison with Baselines. We evaluate our method in two experiment scenarios: (1) In top-K recommendation, we use the trained model to select K items with highest predicted click probability for each user in the test set, and choose *Recall@K* to evaluate the recommended sets. (2) In click-through rate (CTR) prediction, we apply the trained model to predict each piece of user-item pair in the test set (including positive items and randomly selected negative items). We use *AUC* as the evaluation metric in CTR prediction.

The results of top-K recommendation and CTR prediction are presented in Tables 3 and 4, respectively, which show that KGNN-LS outperforms baselines by a significant margin. For example, the *AUC* of KGNN-LS surpasses baselines by 5.1%, 6.9%, 8.3%, and 4.3% on average in MovieLens-20M, Book-Crossing, Last.FM, and Dianping-Food datasets, respectively.

We also show daily performance of KGNN-LS and baselines on Dianping-Food to investigate performance stability. Figure 4 shows their *AUC* score from September 1, 2018 to September 30, 2018. We notice that the curve of KGNN-LS is consistently above baselines over the test period; Moreover, the performance of KGNN-LS is also

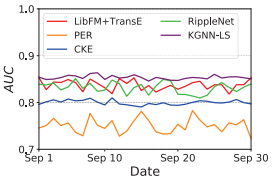


Figure 4: Daily *AUC* of all methods on Dianping-Food in September 2018.

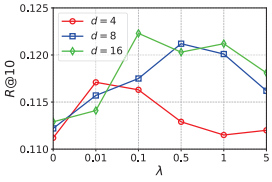


Figure 5: Effectiveness of LS regularization on Last.FM.

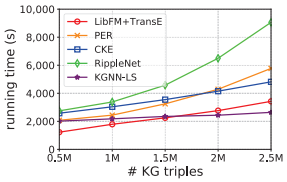


Figure 6: Running time of all methods w.r.t. KG size on MovieLens-20M.

r	20%	40%	60%	80%	100%
SVD	0.882	0.913	0.938	0.955	0.963
LibFM	0.902	0.923	0.938	0.950	0.959
LibFM+TransE	0.914	0.935	0.949	0.960	0.966
PER	0.802	0.814	0.821	0.828	0.832
CKE	0.898	0.910	0.916	0.921	0.924
RippleNet	0.921	0.937	0.947	0.955	0.960
KGNN-LS	**0.961**	**0.970**	**0.974**	**0.977**	**0.979**

Table 5: *AUC* of all methods w.r.t. the ratio of training set *r*.

with low variance, which suggests that KGNN-LS is also robust and stable in practice.

5.4.2 Effectiveness of LS Regularization. Is the proposed LS regularization helpful in improving the performance of GNN? To study the effectiveness of LS regularization, we fix the dimension of hidden layers as 4, 8, and 16, then vary λ from 0 to 5 to see how performance changes. The results of *R*@10 in Last.FM dataset are plotted in Figure 5. It is clear that the performance of KGNN-LS with a non-zero λ is better than $\lambda = 0$ (the case of Wang et al. [28]), which justifies our claim that LS regularization can assist learning the edge weights in a KG and achieve better generalization in recommender systems. But note that a too large λ is less favorable, since it overwhelms the overall loss and misleads the direction of gradients. According to the experiment results, we find that a λ between 0.1 and 1.0 is preferable in most cases.

5.4.3 Results in cold-start scenarios. One major goal of using KGs in recommender systems is to alleviate the sparsity issue. To investigate the performance of KGNN-LS in cold-start scenarios, we vary the size of training set of MovieLens-20M from $r = 100\%$ to $r = 20\%$ (while the validation and test set are kept fixed), and report the results of *AUC* in Table 5. When $r = 20\%$, *AUC* decreases by 8.4%, 5.9%, 5.4%, 3.6%, 2.8%, and 4.1% for the six baselines compared to the model trained on full training data ($r = 100\%$), but the performance decrease of KGNN-LS is only 1.8%. This demonstrates that KGNN-LS still maintains predictive performance even when user-item interactions are sparse.

5.4.4 Hyper-parameters Sensitivity. We first analyze the sensitivity of KGNN-LS to the number of GNN layers *L*. We vary *L* from 1 to 4 while keeping other hyper-parameters fixed. The results are shown in Table 6. We find that the model performs poorly when $L = 4$, which is because a larger *L* will mix too many entity embeddings

L	1	2	3	4
MovieLens-20M	**0.155**	0.146	0.122	0.011
Book-Crossing	0.077	**0.082**	0.043	0.008
Last.FM	**0.122**	0.106	0.105	0.057
Dianping-Food	0.165	**0.170**	0.061	0.036

Table 6: *R*@10 w.r.t. the number of layers *L*.

d	4	8	16	32	64	128
MovieLens-20M	0.134	0.141	0.143	**0.155**	**0.155**	0.151
Book-Crossing	0.065	0.073	0.077	0.081	**0.082**	0.080
Last.FM	0.111	0.116	**0.122**	0.109	0.102	0.107
Dianping-Food	0.155	**0.170**	0.167	0.166	0.163	0.161

Table 7: *R*@10 w.r.t. the dimension of hidden layers *d*.

in a given entity, which *over-smoothes* the representation learning on KGs. KGNN-LS achieves the best performance when $L = 1$ or 2 in the four datasets.

We also examine the impact of the dimension of hidden layers *d* on the performance of KGNN-LS. The result in shown in Table 7. We observe that the performance is boosted with the increase of *d* at the beginning, because more bits in hidden layers can improve the model capacity. However, the performance drops when *d* further increases, since a too large dimension may overfit datasets. The best performance is achieved when $d = 8 \sim 64$.

5.5 Running Time Analysis

We also investigate the running time of our method with respect to the size of KG. We run experiments on a Microsoft Azure virtual machine with 1 NVIDIA Tesla M60 GPU, 12 Intel Xeon CPUs (E5-2690 v3 @2.60GHz), and 128GB of RAM. The size of the KG is increased by up to five times the original one by extracting more triples from Satori, and the running times of all methods on MovieLens-20M are reported in Figure 6. Note that the trend of a curve matters more than the real values, since the values are largely dependent on the minibatch size and the number of epochs (yet we did try to align the configurations of all methods). The result show that KGNN-LS exhibits strong scalability even when the KG is large.

6 CONCLUSION AND FUTURE WORK

In this paper, we propose knowledge-aware graph neural networks with label smoothness regularization for recommendation. KGNN-LS applies GNN architecture to KGs by using user-specific relation

scoring functions and aggregating neighborhood information with different weights. In addition, the proposed label smoothness constraint and leave-one-out loss provide strong regularization for learning the edge weights in KGs. We also discuss how KGs benefit recommender systems and how label smoothness can assist learning the edge weights. Experiment results show that KGNN-LS outperforms state-of-the-art baselines in four recommendation scenarios and achieves desirable scalability with respect to KG size.

In this paper, LS regularization is proposed for recommendation task with KGs. It is interesting to examine the LS assumption on other graph tasks such as link prediction and node classification. Investigating the theoretical relationship between feature propagation and label propagation is also a promising direction.

Acknowledgements. This research has been supported in part by NSF OAC-1835598, DARPA MCS, ARO MURI, Boeing, Docomo, Hitachi, Huawei, JD, Siemens, and Stanford Data Science Initiative.

REFERENCES

[1] Shumeet Baluja, Rohan Seth, D Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly. 2008. Video suggestion and discovery for youtube: taking random walks through the view graph. In *Proceedings of the 17th international conference on World Wide Web*. ACM, 895–904.

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*. 2787–2795.

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral networks and locally connected networks on graphs. In *the 2nd International Conference on Learning Representations*.

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. ACM, 191–198.

[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*. 3844–3852.

[6] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*. 2224–2232.

[7] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[8] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S Yu. 2018. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1531–1540.

[9] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks. In *the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 505–514.

[10] Masayuki Karasuyama and Hiroshi Mamitsuka. 2013. Manifold-based similarity adaptation for label propagation. In *Advances in neural information processing systems*. 1547–1555.

[11] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *the 5th International Conference on Learning Representations*.

[12] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 426–434.

[13] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[14] Federico Monti, Michael Bronstein, and Xavier Bresson. 2017. Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems*. 3697–3707.

[15] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*. 2014–2023.

[16] Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 3 (2012), 57.

[17] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*. Springer, 593–607.

[18] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. 2018. Recurrent knowledge graph embedding for effective recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 297–305.

[19] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. *stat* 1050 (2017), 7.

[20] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the 6th International Conferences on Learning Representations*.

[21] Fei Wang and Changshui Zhang. 2008. Label propagation through linear neighborhoods. *IEEE Transactions on Knowledge and Data Engineering* 20, 1 (2008), 55–67.

[22] Hongwei Wang, Jia Wang, Miao Zhao, Jiannong Cao, and Minyi Guo. 2017. Joint topic-semantic-aware social recommendation for online voting. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 347–356.

[23] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. 2018. Shine: Signed heterogeneous information network embedding for sentiment link prediction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 592–600.

[24] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 417–426.

[25] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Exploring High-Order User Preference on the Knowledge Graph for Recommender Systems. *ACM Transactions on Information Systems (TOIS)* 37, 3 (2019), 32.

[26] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep Knowledge-Aware Network for News Recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. 1835–1844.

[27] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. In *Proceedings of the 2019 World Wide Web Conference on World Wide Web*.

[28] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge graph convolutional networks for recommender systems. In *Proceedings of the 2019 World Wide Web Conference on World Wide Web*.

[29] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 839–848.

[30] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[31] Yuexin Wu, Hanxiao Liu, and Yiming Yang. 2018. Graph Convolutional Matrix Completion for Bipartite Edge Prediction. In *the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*.

[32] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 974–983.

[33] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*. ACM, 283–292.

[34] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 353–362.

[35] Xinhua Zhang and Wee S Lee. 2007. Hyperparameter learning for graph based semi-supervised learning algorithms. In *Advances in neural information processing systems*. 1585–1592.

[36] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-graph based recommendation fusion over heterogeneous information networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 635–644.

[37] Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *Advances in neural information processing systems*. 321–328.

[38] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning*. 912–919.

[39] Xiaojin Zhu, John Lafferty, and Ronald Rosenfeld. 2005. *Semi-supervised learning with graphs*. Ph.D. Dissertation. Carnegie Mellon University, language technologies institute, school of computer science.

APPENDIX

A Additional Details on Datasets

MovieLens-20M, Book-Crossing, and Last.FM datasets contain explicit feedbacks data (Last.FM provides the listening count as weight for each user-item interaction). Therefore, we transform them into implicit feedback, where each entry is marked with 1 indicating that the user has rated the item positively. The threshold of positive rating is 4 for MovieLens-20M, while no threshold is set for Book-Crossing and Last.FM due to their data sparsity. Additionally, we randomly sample an unwatched set of items and mark them as 0 for each user, the number of which equals his/her positively-rated ones.

We use Microsoft Satori to construct the KGs for MovieLens-20M, Book-Crossing, and Last.FM datasets. In each triple of Satori KG, the head and the tail are either IDs or textual content, and the relation follows the format "domain.head_category.tail_category" (e.g., "book.book.author"). We first select a subset of triples from the whole Satori KG with a confidence level greater than 0.9. Then we collect Satori IDs of all valid movies/books/musicians by matching their names with the tail of triples *(head, film.film.name, tail)*, *(head, book.book.title, tail)*, or *(head, type.object.name, tail)*, respectively, for the three datasets. Items with multiple matched or no matched entities are excluded for simplicity. After obtaining the set of item IDs, we match these item IDs with the head of all triples in Satori sub-KG, and select all well-matched triples as the final KG for each dataset.

Dianping-Food dataset is collected from Dianping.com, a Chinese group buying website hosting consumer reviews of restaurants similar to Yelp. We select approximately 10 million interactions between users and restaurants in Dianping.com from May 1, 2015 to December 12, 2018. The types of positive interactions include clicking, buying, and adding to favorites, and we sample negative interactions for each user. The KG for Dianping-Food is collected from Meituan Brain, an internal knowledge graph built for dining and entertainment by Meituan-Dianping Group. The types of entities include POI (restaurant), city, first-level and second-level category, star, business area, dish, and tag; The types of relations correspond to the types of entities (e.g., "organization.POI.has_dish").

	Movie	Book	Music	Restaurant
S	16	8	8	4
d	32	64	16	8
L	1	2	1	2
λ	1.0	0.5	0.1	0.5
γ	10^{-7}	2×10^{-5}	10^{-4}	10^{-7}
η	2×10^{-2}	2×10^{-4}	5×10^{-4}	2×10^{-2}

Table 8: Hyper-parameter settings for the four datasets (S: number of sampled neighbors for each entity; d: dimension of hidden layers, L: number of layers, λ: label smoothness regularizer weight, γ: L2 regularizer weight, η: learning rate).

B Additional Details on Hyper-parameter Searching

In KGNN-LS, we set functions g and f as inner product, σ as *ReLU* for non-last-layers and *tanh* for the last-layer. Note that the number of neighbors of an entity in a KG may be significantly different from each other. Therefore, we uniformly sample a fixed-size set of neighbors for each entity instead of using all of its neighbors to keep the computation more efficient. The number of sampled neighbors for each entity is denoted by S. Hyper-parameter settings for the four datasets are given in Table 8, which are determined by optimizing R@10 on a validation set. The search spaces for hyper-parameters are as follows:

- $S = \{2, 4, 8, 16, 32\}$;
- $d = \{4, 8, 16, 32, 64, 128\}$;
- $L = \{1, 2, 3, 4\}$;
- $\lambda = \{0, 0.01, 0.1, 0.5, 1, 5\}$;
- $\gamma = \{10^{-9}, 10^{-8}, 10^{-7}, 2 \times 10^{-7}, 5 \times 10^{-7}, 10^{-6}, 2 \times 10^{-6}, 5 \times 10^{-6}, 10^{-5}, 2 \times 10^{-5}, 5 \times 10^{-5}, 10^{-4}, 2 \times 10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$;
- $\eta = \{10^{-5}, 2 \times 10^{-5}, 5 \times 10^{-5}, 10^{-4}, 2 \times 10^{-4}, 5 \times 10^{-4}, 10^{-3}, 2 \times 10^{-3}, 5 \times 10^{-3}, 10^{-2}, 2 \times 10^{-2}, 5 \times 10^{-2}, 10^{-1}\}$.

For each dataset, the ratio of training, validation, and test set is $6 : 2 : 2$. Each experiment is repeated 5 times, and the average performance is reported. All trainable parameters are optimized by Adam algorithm.

The Web Conference 2019

Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation

Hongwei Wang[1,2], Fuzheng Zhang[3], Miao Zhao[4], Wenjie Li[4], Xing Xie[2], Minyi Guo[1*]
[1]Shanghai Jiao Tong University, Shanghai, China
[2]Microsoft Research Asia, Beijing, China
[3]Meituan-Dianping Group, Beijing, China
[4]The Hong Kong Polytechnic University, Hong Kong, China
wanghongwei55@gmail.com,zhangfuzheng@meituan.com,{csmiaozhao,cswjli}@comp.polyu.edu.hk
xingx@microsoft.com,guo-my@cs.sjtu.edu.cn

ABSTRACT

Collaborative filtering often suffers from sparsity and cold start problems in real recommendation scenarios, therefore, researchers and engineers usually use side information to address the issues and improve the performance of recommender systems. In this paper, we consider knowledge graphs as the source of side information. We propose **MKR**, a **M**ulti-task feature learning approach for **K**nowledge graph enhanced **R**ecommendation. MKR is a deep end-to-end framework that utilizes knowledge graph embedding task to assist recommendation task. The two tasks are associated by cross&compress units, which automatically share latent features and learn high-order interactions between items in recommender systems and entities in the knowledge graph. We prove that cross&compress units have sufficient capability of polynomial approximation, and show that MKR is a generalized framework over several representative methods of recommender systems and multi-task learning. Through extensive experiments on real-world datasets, we demonstrate that MKR achieves substantial gains in movie, book, music, and news recommendation, over state-of-the-art baselines. MKR is also shown to be able to maintain a decent performance even if user-item interactions are sparse.

KEYWORDS

Recommender systems; knowledge graph; multi-task learning

ACM Reference Format:
Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation In *Proceedings of The 2019 Web Conference (WWW 2019).* ACM, New York, NY, USA, 11 pages. https://doi.org/xxxxx

1 INTRODUCTION

Recommender systems (RS) aims to address the information explosion and meet users personalized interests. One of the most popular recommendation techniques is collaborative filtering (CF) [11], which utilizes users' historical interactions and makes recommendations based on their common preferences. However, CF-based methods usually suffer from the sparsity of user-item interactions and the cold start problem. Therefore, researchers propose using *side information* in recommender systems, including social networks [10], attributes [30], and multimedia (e.g., texts [29], images [40]). *Knowledge graphs* (KGs) are one type of side information for RS, which usually contain fruitful facts and connections about items. Recently, researchers have proposed several academic and commercial KGs, such as NELL[1], DBpedia[2], Google Knowledge Graph[3] and Microsoft Satori[4]. Due to its high dimensionality and heterogeneity, a KG is usually pre-processed by *knowledge graph embedding* (KGE) methods [27], which embeds entities and relations into low-dimensional vector spaces while preserving its inherent structure.

Existing KG-aware methods

Inspired by the success of applying KG in a wide variety of tasks, researchers have recently tried to utilize KG to improve the performance of recommender systems [31, 32, 39, 40, 45]. Personalized Entity Recommendation (PER) [39] and Factorization Machine with Group lasso (FMG) [45] treat KG as a heterogeneous information network, and extract meta-path/meta-graph based latent features to represent the connectivity between users and items along different types of relation paths/graphs. It should be noted that PER and FMG rely heavily on manually designed meta-paths/meta-graphs, which limits its application in generic recommendation scenarios. Deep Knowledge-aware Network (DKN) [32] designs a CNN framework to combine entity embeddings with word embeddings for news recommendation. However, the entity embeddings are required in advance of using DKN, causing DKN to lack an end-to-end way of training. Another concern about DKN is that it can hardly incorporate side information other than texts. RippleNet [31] is a memory-network-like model that propagates users' potential preferences in the KG and explores their hierarchical interests. But the importance of relations is weakly characterized in RippleNet, because the embedding matrix of a relation $\mathbf{R}$ can hardly be trained to capture the sense of importance in the quadratic form $\mathbf{v}^\top \mathbf{R}\mathbf{h}$ ($\mathbf{v}$ and $\mathbf{h}$ are embedding vectors of two entities). Collaborative Knowledge base Embedding (CKE) [40] combines CF with structural knowledge, textual knowledge, and visual knowledge in a unified framework. However, the KGE module in CKE (i.e., TransR [13]) is more suitable for in-graph applications (such as KG completion and link prediction) rather than recommendation. In addition, the CF module and the KGE module are loosely coupled

*M. Guo is the corresponding author. This work was partially sponsored by the National Basic Research 973 Program of China under Grant 2015CB352403.

WWW 2019, May 13–17, 2019, San Francisco, USA
2019. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[1]http://rtw.ml.cmu.edu/rtw/
[2]http://wiki.dbpedia.org/
[3]https://developers.google.com/knowledge-graph/
[4]https://searchengineland.com/library/bing/bing-satori

in CKE under a Bayesian framework, making the supervision from KG less obvious for recommender systems.

The proposed approach

To address the limitations of previous work, we propose MKR, a multi-task learning (MTL) approach for knowledge graph enhanced recommendation. MKR is a generic, end-to-end deep recommendation framework, which aims to utilize KGE task to assist recommendation task[5]. Note that the two tasks are not mutually independent, but are highly correlated since an item in RS may associate with one or more entities in KG. Therefore, an item and its corresponding entity are likely to have a similar proximity structure in RS and KG, and share similar features in low-level and non-task-specific latent feature spaces [15]. We will further validate the similarity in the experiments section. To model the shared features between items and entities, we design a *cross&compress unit* in MKR. The cross&compress unit explicitly models high-order interactions between item and entity features, and automatically control the cross knowledge transfer for both tasks. Through cross&compress units, representations of items and entities can complement each other, assisting both tasks in avoiding fitting noises and improving generalization. The whole framework can be trained by alternately optimizing the two tasks with different frequencies, which endows MKR with high flexibility and adaptability in real recommendation scenarios.

We probe the expressive capability of MKR and show, through theoretical analysis, that the cross&compress unit is capable of approximating sufficiently high order feature interactions between items and entities. We also show that MKR is a generalized framework over several representative methods of recommender systems and multi-task learning, including factorization machines [22, 23], deep&cross network [34], and cross-stitch network [18]. Empirically, we evaluate our method in four recommendation scenarios, i.e., movie, book, music, and news recommendations. The results demonstrate that MKR achieves substantial gains over state-of-the-art baselines in both click-through rate (CTR) prediction (e.g., 11.6% *AUC* improvements on average for movies) and top-K recommendation (e.g., 66.4% *Recall*@10 improvements on average for books). MKR can also maintain a decent performance in sparse scenarios.

Contribution

It is worth noticing that the problem studied in this paper can also be modelled as *cross-domain recommendation* [26] or *transfer learning* [21], since we care more about the performance of recommendation task. However, the key observation is that though cross-domain recommendation and transfer learning have single objective for the target domain, their loss functions still contain constraint terms for measuring data distribution in the source domain or similarity between two domains. In our proposed MKR, the KGE task serves as the constraint term *explicitly* to provide regularization for recommender systems. We would like to emphasize that the major contribution of this paper is exactly modeling the problem as multi-task learning: We go a step further than cross-domain recommendation and transfer learning by finding that the inter-task similarity is helpful to not only recommender systems but also knowledge graph embedding, as shown in theoretical analysis and experiment results.

[5]KGE task can also benefit from recommendation task empirically as shown in the experiments section.

2 OUR APPROACH

In this section, we first formulate the knowledge graph enhanced recommendation problem, then introduce the framework of MKR and present the design of the cross&compress unit, recommendation module and KGE module in detail. We lastly discuss the learning algorithm for MKR.

2.1 Problem Formulation

We formulate the knowledge graph enhanced recommendation problem in this paper as follows. In a typical recommendation scenario, we have a set of M users $\mathcal{U} = \{u_1, u_2, ..., u_M\}$ and a set of N items $\mathcal{V} = \{v_1, v_2, ..., v_N\}$. The user-item interaction matrix $\mathbf{Y} \in \mathbb{R}^{M \times N}$ is defined according to users' implicit feedback, where $y_{uv} = 1$ indicates that user u engaged with item v, such as behaviors of clicking, watching, browsing, or purchasing; otherwise $y_{uv} = 0$. Additionally, we also have access to a knowledge graph $\mathcal{G}$, which is comprised of entity-relation-entity triples (h, r, t). Here h, r, and t denote the head, relation, and tail of a knowledge triple, respectively. For example, the triple (*Quentin Tarantino, film.director.film, Pulp Fiction*) states the fact that Quentin Tarantino directs the film Pulp Fiction. In many recommendation scenarios, an item $v \in \mathcal{V}$ may associate with one or more entities in $\mathcal{G}$. For example, in movie recommendation, the item "Pulp Fiction" is linked with its namesake in a KG, while in news recommendation, news with the title "Trump pledges aid to Silicon Valley during tech meeting" is linked with entities "Donald Trump" and "Silicon Valley" in a KG.

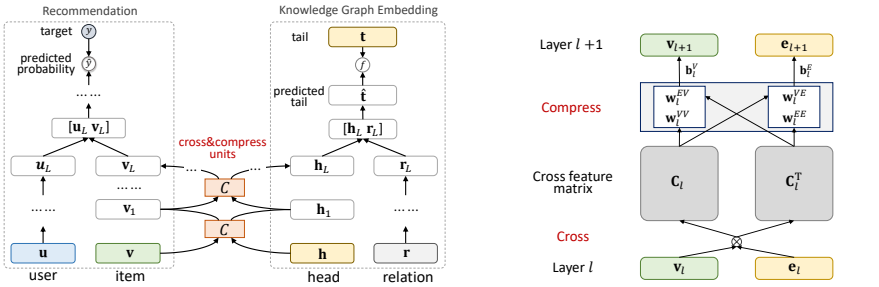
Given the user-item interaction matrix $\mathbf{Y}$ as well as the knowledge graph $\mathcal{G}$, we aim to predict whether user u has potential interest in item v with which he has had no interaction before. Our goal is to learn a prediction function $\hat{y}_{uv} = \mathcal{F}(u, v|\Theta, \mathbf{Y}, \mathcal{G})$, where $\hat{y}_{uv}$ denotes the probability that user u will engage with item v, and Θ is the model parameters of function $\mathcal{F}$.

2.2 Framework

The framework of MKR is illustrated in Figure 1a. MKR consists of three main components: recommendation module, KGE module, and cross&compress units. (1) The recommendation module on the left takes a user and an item as input, and uses a multi-layer perceptron (MLP) and cross&compress units to extract short and dense features for the user and the item, respectively. The extracted features are then fed into another MLP together to output the predicted probability. (2) Similar to the left part, the KGE module in the right part also uses multiple layers to extract features from the head and relation of a knowledge triple, and outputs the representation of the predicted tail under the supervision of a score function f and the real tail. (3) The recommendation module and the KGE module are bridged by specially designed cross&compress units. The proposed unit can automatically learn high-order feature interactions of items in recommender systems and entities in the KG.

2.3 Cross&compress Unit

To model feature interactions between items and entities, we design a cross&compress unit in MKR framework. As shown in Figure 1b,



(a) Framework of MKR

(b) Cross&compress unit

Figure 1: (a) The framework of MKR. The left and right part illustrate the recommendation module and the KGE module, respectively, which are bridged by the cross&compress units. (b) Illustration of a cross&compress unit. The cross&compress unit generates a cross feature matrix from item and entity vectors by *cross* operation, and outputs their vectors for the next layer by *compress* operation.

for item v and one of its associated entities e, we first construct $d \times d$ pairwise interactions of their latent feature $\mathbf{v}_l \in \mathbb{R}^d$ and $\mathbf{e}_l \in \mathbb{R}^d$ from layer l:

$$\mathbf{C}_l = \mathbf{v}_l \mathbf{e}_l^\top = \begin{bmatrix} v_l^{(1)} e_l^{(1)} & \cdots & v_l^{(1)} e_l^{(d)} \\ \cdots & & \cdots \\ v_l^{(d)} e_l^{(1)} & \cdots & v_l^{(d)} e_l^{(d)} \end{bmatrix}, \tag{1}$$

where $\mathbf{C}_l \in \mathbb{R}^{d \times d}$ is the cross feature matrix of layer l, and d is the dimension of hidden layers. This is called the *cross* operation, since each possible feature interaction $v_l^{(i)} e_l^{(j)}, \forall (i,j) \in \{1, ..., d\}^2$ between item v and its associated entity e is modeled explicitly in the cross feature matrix. We then output the feature vectors of items and entities for the next layer by projecting the cross feature matrix into their latent representation spaces:

$$\begin{aligned} \mathbf{v}_{l+1} &= \mathbf{C}_l \mathbf{w}_l^{VV} + \mathbf{C}_l^\top \mathbf{w}_l^{EV} + \mathbf{b}_l^V = \mathbf{v}_l \mathbf{e}_l^\top \mathbf{w}_l^{VV} + \mathbf{e}_l \mathbf{v}_l^\top \mathbf{w}_l^{EV} + \mathbf{b}_l^V, \\ \mathbf{e}_{l+1} &= \mathbf{C}_l \mathbf{w}_l^{VE} + \mathbf{C}_l^\top \mathbf{w}_l^{EE} + \mathbf{b}_l^E = \mathbf{v}_l \mathbf{e}_l^\top \mathbf{w}_l^{VE} + \mathbf{e}_l \mathbf{v}_l^\top \mathbf{w}_l^{EE} + \mathbf{b}_l^E, \end{aligned} \tag{2}$$

where $\mathbf{w}_l^{\cdot\cdot} \in \mathbb{R}^d$ and $\mathbf{b}_l^{\cdot} \in \mathbb{R}^d$ are trainable weight and bias vectors. This is called the *compress* operation, since the weight vectors project the cross feature matrix from $\mathbb{R}^{d \times d}$ space back to the feature spaces $\mathbb{R}^d$. Note that in Eq. (2), the cross feature matrix is compressed along both horizontal and vertical directions (by operating on $\mathbf{C}_l$ and $\mathbf{C}_l^\top$) for the sake of symmetry, but we will provide more insights of the design in Section 3.2. For simplicity, the cross&compress unit is denoted as:

$$[\mathbf{v}_{l+1}, \mathbf{e}_{l+1}] = C(\mathbf{v}_l, \mathbf{e}_l), \tag{3}$$

and we use a suffix $[\mathbf{v}]$ or $[\mathbf{e}]$ to distinguish its two outputs in the following of this paper. Through cross&compress units, MKR can adaptively adjust the weights of knowledge transfer and learn the relevance between the two tasks.

It should be noted that cross&compress units should only exist in low-level layers of MKR, as shown in Figure 1a. This is because: (1) In deep architectures, features usually transform from general to specific along the network, and feature transferability drops significantly in higher layers with increasing task dissimilarity [38]. Therefore, sharing high-level layers risks to possible negative transfer, especially for the heterogeneous tasks in MKR. (2) In high-level layers of MKR, item features are mixed with user features, and entity features are mixed with relation features. The mixed features are not suitable for sharing since they have no explicit association.

2.4 Recommendation Module

The input of the recommendation module in MKR consists of two raw feature vectors $\mathbf{u}$ and $\mathbf{v}$ that describe user u and item v, respectively. $\mathbf{u}$ and $\mathbf{v}$ can be customized as one-hot ID [8], attributes [30], bag-of-words [29], or their combinations, based on the application scenario. Given user u's raw feature vector $\mathbf{u}$, we use an L-layer MLP to extract his latent condensed feature[6]:

$$\mathbf{u}_L = \mathcal{M}(\mathcal{M}(\cdots \mathcal{M}(\mathbf{u}))) = \mathcal{M}^L(\mathbf{u}), \tag{4}$$

where $\mathcal{M}(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x}+\mathbf{b})$ is a fully-connected neural network layer[7] with weight $\mathbf{W}$, bias $\mathbf{b}$, and nonlinear activation function $\sigma(\cdot)$. For item v, we use L cross&compress units to extract its feature:

$$\mathbf{v}_L = \mathbb{E}_{e \sim \mathcal{S}(v)} \left[C^L(\mathbf{v}, \mathbf{e})[\mathbf{v}] \right], \tag{5}$$

where $\mathcal{S}(v)$ is the set of associated entities of item v.

After having user u's latent feature $\mathbf{u}_L$ and item v's latent feature $\mathbf{v}_L$, we combine the two pathways by a predicting function f_{RS}, for example, inner product or an H-layer MLP. The final predicted probability of user u engaging item v is:

$$\hat{y}_{uv} = \sigma\big(f_{RS}(\mathbf{u}_L, \mathbf{v}_L)\big). \tag{6}$$

[6] We use the exponent notation L in Eq. (4) and following equations in the rest of this paper for simplicity, but note that the parameters of L layers are actually different.
[7] Exploring a more elaborate design of layers in the recommendation module is an important direction of future work.

2.5 Knowledge Graph Embedding Module

Knowledge graph embedding is to embed entities and relations into continuous vector spaces while preserving their structure. Recently, researchers have proposed a great many KGE methods, including translational distance models [2, 13] and semantic matching models [14, 19]. In MKR, we propose a deep semantic matching architecture for KGE module. Similar to the recommendation module, for a given knowledge triple (h, r, t), we first utilize multiple cross&compress units and nonlinear layers to process the raw feature vectors of head h and relation r (including ID [13], types [36], textual description [35], etc.), respectively. Their latent features are then concatenated together, followed by a K-layer MLP for predicting tail t:

$$\begin{aligned} \mathbf{h}_L &= \mathbb{E}_{v \sim \mathcal{S}(h)} \left[C^L(\mathbf{v}, \mathbf{h})[\mathbf{e}] \right], \\ \mathbf{r}_L &= \mathcal{M}^L(\mathbf{r}), \\ \hat{\mathbf{t}} &= \mathcal{M}^K \left(\begin{bmatrix} \mathbf{h}_L \\ \mathbf{r}_L \end{bmatrix} \right), \end{aligned} \tag{7}$$

where $\mathcal{S}(h)$ is the set of associated items of entity h, and $\hat{\mathbf{t}}$ is the predicted vector of tail t. Finally, the score of the triple (h, r, t) is calculated using a score (similarity) function f_{KG}:

$$score(h, r, t) = f_{KG}(\mathbf{t}, \hat{\mathbf{t}}), \tag{8}$$

where $\mathbf{t}$ is the real feature vector of t. In this paper, we use the normalized inner product $f_{KG}(\mathbf{t}, \hat{\mathbf{t}}) = \sigma(\mathbf{t}^\top \hat{\mathbf{t}})$ as the choice of score function [18], but other forms of (dis)similarity metrics can also be applied here such as KullbackâĂŞLeibler divergence.

2.6 Learning Algorithm

The complete loss function of MKR is as follows:

$$\begin{aligned} \mathcal{L} =& \mathcal{L}_{RS} + \mathcal{L}_{KG} + \mathcal{L}_{REG} \\ =& \sum_{u \in \mathcal{U}, v \in \mathcal{V}} \mathcal{J}(\hat{y}_{uv}, y_{uv}) \\ & - \lambda_1 \Big(\sum_{(h,r,t) \in \mathcal{G}} score(h, r, t) - \sum_{(h',r,t') \notin \mathcal{G}} score(h', r, t') \Big) \\ & + \lambda_2 \|\mathbf{W}\|_2^2. \end{aligned} \tag{9}$$

In Eq. (9), the first term measures loss in the recommendation module, where u and v traverse the set of users and the items, respectively, and $\mathcal{J}$ is the cross-entropy function. The second term calculates the loss in the KGE module, in which we aim to increase the score for all true triples while reducing the score for all false triples. The last item is the regularization term for preventing overfitting, λ_1 and λ_2 are the balancing parameters.[8]

Note that the loss function in Eq. (9) traverses all possible user-item pairs and knowledge triples. To make computation more efficient, following [17], we use a negative sampling strategy during training. The learning algorithm of MKR is presented in Algorithm 1, in which a training epoch consists of two stages: recommendation task (line 3-7) and KGE task (line 8-10). In each iteration, we repeat training on recommendation task for t times (t is a hyper-parameter and normally $t > 1$) before training on KGE task once in each epoch, since we are more focused on improving recommendation performance. We will discuss the choice of t in the experiments section.

[8] λ_1 can be seen as the ratio of two learning rates for the two tasks.

Algorithm 1 Multi-Task Training for MKR

Input: Interaction matrix $\mathbf{Y}$, knowledge graph $\mathcal{G}$
Output: Prediction function $\mathcal{F}(u, v|\Theta, \mathbf{Y}, \mathcal{G})$

1: Initialize all parameters
2: **for** number of training iteration **do**
   // *recommendation task*
3: **for** t steps **do**
4: Sample minibatch of positive and negative interactions from $\mathbf{Y}$;
5: Sample $e \sim \mathcal{S}(v)$ for each item v in the minibatch;
6: Update parameters of $\mathcal{F}$ by gradient descent on Eq. (1)-(6), (9);
7: **end for**
   // *knowledge graph embedding task*
8: Sample minibatch of true and false triples from $\mathcal{G}$;
9: Sample $v \sim \mathcal{S}(h)$ for each head h in the minibatch;
10: Update parameters of $\mathcal{F}$ by gradient descent on Eq. (1)-(3), (7)-(9);
11: **end for**

3 THEORETICAL ANALYSIS

In this section, we prove that cross&compress units have sufficient capability of polynomial approximation. We also show that MKR is a generalized framework over several representative methods of recommender systems and multi-task learning.

3.1 Polynomial Approximation

According to the Weierstrass approximation theorem [25], any function under certain smoothness assumption can be approximated by a polynomial to an arbitrary accuracy. Therefore, we examine the ability of high-order interaction approximation of the cross&compress unit. We show that cross&compress units can model the order of item-entity feature interaction up to exponential degree:

Theorem 1. *Denote the input of item and entity in MKR network as* $\mathbf{v} = [v_1 \cdots v_d]^\top$ *and* $\mathbf{e} = [e_1 \cdots e_d]^\top$*, respectively. Then the cross terms about* $\mathbf{v}$ *and* $\mathbf{e}$ *in* $\|\mathbf{v}_L\|_1$ *and* $\|\mathbf{e}_L\|_1$ *(the L1-norm of* $\mathbf{v}_L$ *and* $\mathbf{e}_L$*) with maximal degree is* $k_{\boldsymbol{\alpha},\boldsymbol{\beta}} v_1^{\alpha_1} \cdots v_d^{\alpha_d} e_1^{\beta_1} \cdots e_d^{\beta_d}$*, where* $k_{\boldsymbol{\alpha},\boldsymbol{\beta}} \in \mathbb{R}$, $\alpha_i, \beta_i \in \mathbb{N}$ *for* $i \in \{1, \cdots, d\}$, $\alpha_1 + \cdots + \alpha_d = 2^{L-1}$*, and* $\beta_1 + \cdots + \beta_d = 2^{L-1}$ *(*$L \geq 1$, $\mathbf{v}_0 = \mathbf{v}$, $\mathbf{e}_0 = \mathbf{e}$*).*

In recommender systems, $\prod_{i=1}^{d} v_i^{\alpha_i} e_i^{\beta_i}$ is also called *combinatorial* feature, as it measures the interactions of multiple original features. Theorem 1 states that cross&compress units can automatically model the combinatorial features of items and entities for sufficiently high order, which demonstrates the superior approximation capacity of MKR as compared with existing work such as Wide&Deep [3], factorization machines [22, 23] and DCN [34]. The proof of Theorem 1 is provided in the Appendix. Note that Theorem 1 gives a theoretical view of the polynomial approximation

ability of the cross&compress unit rather than providing guarantees on its actual performance. We will empirically evaluate the cross&compress unit in the experiments section.

3.2 Unified View of Representative Methods

In the following we provide a unified view of several representative models in recommender systems and multi-task learning, by showing that they are restricted versions of or theoretically related to MKR. This justifies the design of cross&compress unit and conceptually explains its strong empirical performance as compared to baselines.

3.2.1 Factorization machines. Factorization machines [22, 23] are a generic method for recommender systems. Given an input feature vector, FMs model all interactions between variables in the input vector using factorized parameters, thus being able to estimate interactions in problems with huge sparsity such as recommender systems. The model equation for a 2-degree factorization machine is defined as

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j, \quad (10)$$

where x_i is the i-th unit of input vector $\mathbf{x}$, $w_\cdot$ is weight scalar, $\mathbf{v}_\cdot$ is weight vector, and $\langle \cdot, \cdot \rangle$ is dot product of two vectors. We show that the essence of FM is conceptually similar to an 1-layer cross&compress unit:

Proposition 1. *The L1-norm of $\mathbf{v}_1$ and $\mathbf{e}_1$ can be written as the following form:*

$$\|\mathbf{v}_1\|_1 \ (or \ \|\mathbf{e}_1\|_1) = \left| b + \sum_{i=1}^{d} \sum_{j=1}^{d} \langle w_i, w_j \rangle v_i e_j \right|, \quad (11)$$

where $\langle w_i, w_j \rangle = w_i + w_j$ is the sum of two scalars.

It is interesting to notice that, instead of factorizing the weight parameter of $x_i x_j$ into the dot product of two vectors as in FM, the weight of term $v_i e_j$ is factorized into the sum of two scalars in cross&compress unit to reduce the number of parameters and increase robustness of the model.

3.2.2 Deep&Cross Network. DCN [34] learns explicit and high-order cross features by introducing the layers:

$$\mathbf{x}_{l+1} = \mathbf{x}_0 \mathbf{x}_l^\top \mathbf{w}_l + \mathbf{x}_l + \mathbf{b}_l, \quad (12)$$

where $\mathbf{x}_l$, $\mathbf{w}_l$, and $\mathbf{b}_l$ are representation, weight, and bias of the l-th layer. We demonstrate the link between DCN and MKR by the following proposition:

Proposition 2. *In the formula of $\mathbf{v}_{l+1}$ in Eq. (2), if we restrict $\mathbf{w}_l^{VV}$ in the first term to satisfy $\mathbf{e}_l^\top \mathbf{w}_l^{VV} = 1$ and restrict $\mathbf{e}_l$ in the second term to be $\mathbf{e}_0$ (and impose similar restrictions on $\mathbf{e}_{l+1}$), the cross&compress unit is then conceptually equivalent to DCN layer in the sense of multi-task learning:*

$$\begin{aligned} \mathbf{v}_{l+1} &= \mathbf{e}_0 \mathbf{v}_l^\top \mathbf{w}_l^{EV} + \mathbf{v}_l + \mathbf{b}_l^V, \\ \mathbf{e}_{l+1} &= \mathbf{v}_0 \mathbf{e}_l^\top \mathbf{w}_l^{VE} + \mathbf{e}_l + \mathbf{b}_l^E. \end{aligned} \quad (13)$$

It can be proven that the polynomial approximation ability of the above DCN-equivalent version (i.e., the maximal degree of cross terms in $\mathbf{v}_l$ and $\mathbf{e}_l$) is $O(l)$, which is weaker than original cross&compress units with $O(2^l)$ approximation ability.

3.2.3 Cross-stitch Networks. Cross-stitch networks [18] is a multi-task learning model in convolutional networks, in which the designed cross-stitch unit can learn a combination of shared and task-specific representations between two tasks. Specifically, given two activation maps x_A and x_B from layer l for both the tasks, cross-stitch networks learn linear combinations $\tilde{x}_A$ and $\tilde{x}_B$ of both the input activations and feed these combinations as input to the next layers' filters. The formula at location (i, j) in the activation map is

$$\begin{bmatrix} \tilde{x}_A^{ij} \\ \tilde{x}_B^{ij} \end{bmatrix} = \begin{bmatrix} \alpha_{AA} & \alpha_{AB} \\ \alpha_{BA} & \alpha_{BB} \end{bmatrix} \begin{bmatrix} x_A^{ij} \\ x_B^{ij} \end{bmatrix}, \quad (14)$$

where α's are trainable transfer weights of representations between task A and task B. We show that the cross-stitch unit in Eq. (14) is a simplified version of our cross&compress unit by the following proposition:

Proposition 3. *If we omit all biases in Eq. (2), the cross&compress unit can be written as*

$$\begin{bmatrix} \mathbf{v}_{l+1} \\ \mathbf{e}_{l+1} \end{bmatrix} = \begin{bmatrix} \mathbf{e}_l^\top \mathbf{w}_l^{VV} & \mathbf{v}_l^\top \mathbf{w}_l^{EV} \\ \mathbf{e}_l^\top \mathbf{w}_l^{VE} & \mathbf{v}_l^\top \mathbf{w}_l^{EE} \end{bmatrix} \begin{bmatrix} \mathbf{v}_l \\ \mathbf{e}_l \end{bmatrix}. \quad (15)$$

The transfer matrix in Eq. (15) serves as the cross-stitch unit $[\alpha_{AA} \ \alpha_{AB}; \ \alpha_{BA} \ \alpha_{BB}]$ in Eq. (14). Like cross-stitch networks, MKR network can decide to make certain layers task specific by setting $\mathbf{v}_l^\top \mathbf{w}_l^{EV}$ (α_{AB}) or $\mathbf{e}_l^\top \mathbf{w}_l^{VE}$ (α_{BA}) to zero, or choose a more shared representation by assigning a higher value to them. But the transfer matrix is more fine-grained in cross&compress unit, because the transfer weights are replaced from scalars to dot products of two vectors. It is rather interesting to notice that Eq. (15) can also be regarded as an *attention mechanism* [1], as the computation of transfer weights involves the feature vectors $\mathbf{v}_l$ and $\mathbf{e}_l$ themselves.

4 EXPERIMENTS

In this section, we evaluate the performance of MKR in four real-world recommendation scenarios: movie, book, music, and news[9].

4.1 Datasets

We utilize the following four datasets in our experiments:

- **MovieLens-1M**[10] is a widely used benchmark dataset in movie recommendations, which consists of approximately 1 million explicit ratings (ranging from 1 to 5) on the MovieLens website.
- **Book-Crossing**[11] dataset contains 1,149,780 explicit ratings (ranging from 0 to 10) of books in the Book-Crossing community.
- **Last.FM**[12] dataset contains musician listening information from a set of 2 thousand users from Last.fm online music system.
- **Bing-News** dataset contains 1,025,192 pieces of implicit feedback collected from the server logs of Bing News[13] from

[9] The source code is available at https://github.com/hwwang55/MKR.
[10] https://grouplens.org/datasets/movielens/1m/
[11] http://www2.informatik.uni-freiburg.de/~cziegler/BX/
[12] https://grouplens.org/datasets/hetrec-2011/
[13] https://www.bing.com/news

Table 1: Basic statistics and hyper-parameter settings for the four datasets.

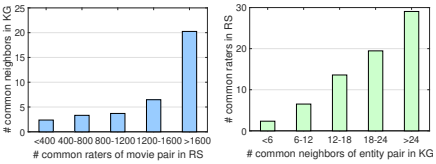
Dataset	# users	# items	# interactions	# KG triples	Hyper-parameters
MovieLens-1M	6,036	2,347	753,772	20,195	$L = 1, d = 8, t = 3, \lambda_1 = 0.5$
Book-Crossing	17,860	14,910	139,746	19,793	$L = 1, d = 8, t = 2, \lambda_1 = 0.1$
Last.FM	1,872	3,846	42,346	15,518	$L = 2, d = 4, t = 2, \lambda_1 = 0.1$
Bing-News	141,487	535,145	1,025,192	1,545,217	$L = 3, d = 16, t = 5, \lambda_1 = 0.2$

October 16, 2016 to August 11, 2017. Each piece of news has a title and a snippet.

Since MovieLens-1M, Book-Crossing, and Last.FM are explicit feedback data (Last.FM provides the listening count as weight for each user-item interaction), we transform them into implicit feedback where each entry is marked with 1 indicating that the user has rated the item positively, and sample an unwatched set marked as 0 for each user. The threshold of positive rating is 4 for MovieLens-1M, while no threshold is set for Book-Crossing and Last.FM due to their sparsity.

We use Microsoft Satori to construct the KG for each dataset. We first select a subset of triples from the whole KG with a confidence level greater than 0.9. For MovieLens-1M and Book-Crossing, we additionally select a subset of triples from the sub-KG whose relation name contains "film" or "book" respectively to further reduce KG size.

Given the sub-KGs, for MovieLens-1M, Book-Crossing, and Last.FM, we collect IDs of all valid movies, books, or musicians by matching their names with tail of triples (*head, film.film.name, tail*), *(head, book.book.title, tail)*, or *(head, type.object.name, tail)*, respectively. For simplicity, items with no matched or multiple matched entities are excluded. We then match the IDs with the head and tail of all KG triples and select all well-matched triples from the sub-KG. The constructing process is similar for Bing-News except that: (1) we use entity linking tools to extract entities in news titles; (2) we do not impose restrictions on the names of relations since the entities in news titles are not within one particular domain. The basic statistics of the four datasets are presented in Table 1. Note that the number of users, items, and interactions are smaller than original datasets since we filtered out items with no corresponding entity in the KG.



(a) RS to KG **(b) KG to RS**

Figure 2: The correlation between the number of common neighbors of an item pair in KG and their number of common raters in RS.

4.2 Baselines

We compare our proposed MKR with the following baselines. Unless otherwise specified, the hyper-parameter settings of baselines are the same as reported in their original papers or as default in their codes.

- **PER** [39] treats the KG as heterogeneous information networks and extracts meta-path based features to represent the connectivity between users and items. In this paper, we use manually designed user-item-attribute-item paths as features, i.e., "user-movie-director-movie", "user-movie-genre-movie", and "user-movie-star-movie" for MovieLens-20M; "user-book-author-book" and "user-book-genre-book" for Book-Crossing; "user-musician-genre-musician", "user-musician-country-musician", and "user-musician-age-musician" (age is discretized) for Last.FM. Note that PER cannot be applied to news recommendation because it's hard to pre-define meta-paths for entities in news.
- **CKE** [40] combines CF with structural, textual, and visual knowledge in a unified framework for recommendation. We implement CKE as CF plus structural knowledge module in this paper. The dimension of user and item embeddings for the four datasets are set as 64, 128, 32, 64, respectively. The dimension of entity embeddings is 32.
- **DKN** [32] treats entity embedding and word embedding as multiple channels and combines them together in CNN for CTR prediction. In this paper, we use movie/book names and news titles as textual input for DKN. The dimension of word embedding and entity embedding is 64, and the number of filters is 128 for each window size 1, 2, 3.
- **RippleNet** [31] is a memory-network-like approach that propagates usersâĂŹ preferences on the knowledge graph for recommendation. The hyper-parameter settings for Last.FM are $d = 8$, $H = 2$, $\lambda_1 = 10^{-6}$, $\lambda_2 = 0.01$, $\eta = 0.02$.
- **LibFM** [23] is a widely used feature-based factorization model. We concatenate the raw features of users and items as well as the corresponding averaged entity embeddings learned from TransR [13] as input for LibFM. The dimension is {1, 1, 8} and the number of training epochs is 50. The dimension of TransR is 32.
- **Wide&Deep** [3] is a deep recommendation model combining a (wide) linear channel with a (deep) nonlinear channel. The input for Wide&Deep is the same as in LibFM. The dimension of user, item, and entity is 64, and we use a two-layer deep channel with dimension of 100 and 50 as well as a wide channel.

Table 2: The results of *AUC* and *Accuracy* in CTR prediction.

Model	MovieLens-1M		Book-Crossing		Last.FM		Bing-News	
	AUC	*ACC*	*AUC*	*ACC*	*AUC*	*ACC*	*AUC*	*ACC*
PER	0.710 (-22.6%)	0.664 (-21.2%)	0.623 (-15.1%)	0.588 (-16.7%)	0.633 (-20.6%)	0.596 (-20.7%)	-	-
CKE	0.801 (-12.6%)	0.742 (-12.0%)	0.671 (-8.6%)	0.633 (-10.3%)	0.744 (-6.6%)	0.673 (-10.5%)	0.553 (-19.7%)	0.516 (-20.0%)
DKN	0.655 (-28.6%)	0.589 (-30.1%)	0.622 (-15.3%)	0.598 (-15.3%)	0.602 (-24.5%)	0.581 (-22.7%)	0.667 (-3.2%)	0.610 (-5.4%)
RippleNet	**0.920** (+0.3%)	0.842 (-0.1%)	0.729 (-0.7%)	0.662 (-6.2%)	0.768 (-3.6%)	0.691 (-8.1%)	0.678 (-1.6%)	0.630 (-2.3%)
LibFM	0.892 (-2.7%)	0.812 (-3.7%)	0.685 (-6.7%)	0.640 (-9.3%)	0.777 (-2.5%)	0.709 (-5.7%)	0.640 (-7.1%)	0.591 (-8.4%)
Wide&Deep	0.898 (-2.1%)	0.820 (-2.7%)	0.712 (-3.0%)	0.624 (-11.6%)	0.756 (-5.1%)	0.688 (-8.5%)	0.651 (-5.5%)	0.597 (-7.4%)
MKR	0.917	**0.843**	**0.734**	**0.704**	**0.797**	**0.752**	**0.689**	**0.645**
MKR-1L	-	-	-	-	0.795 (-0.3%)	0.749 (-0.4%)	0.680 (-1.3%)	0.631 (-2.2%)
MKR-DCN	0.883 (-3.7%)	0.802 (-4.9%)	0.705 (-4.3%)	0.676 (-4.2%)	0.778 (-2.4%)	0.730 (-2.9%)	0.671 (-2.6%)	0.614 (-4.8%)
MKR-stitch	0.905 (-1.3%)	0.830 (-1.5%)	0.721 (-2.2%)	0.682 (-3.4%)	0.772 (-3.1%)	0.725 (-3.6%)	0.674 (-2.2%)	0.621 (-3.7%)

4.3 Experiments setup

In MKR, we set the number of high-level layers $K = 1$, f_{RS} as inner product, and $\lambda_2 = 10^{-6}$ for all three datasets, and other hyper-parameter are given in Table 1. The settings of hyper-parameters are determined by optimizing *AUC* on a validation set. For each dataset, the ratio of training, validation, and test set is $6:2:2$. Each experiment is repeated 3 times, and the average performance is reported. We evaluate our method in two experiment scenarios: (1) In click-through rate (CTR) prediction, we apply the trained model to each piece of interactions in the test set and output the predicted click probability. We use *AUC* and *Accuracy* to evaluate the performance of CTR prediction. (2) In top-K recommendation, we use the trained model to select K items with highest predicted click probability for each user in the test set, and choose *Precision@K* and *Recall@K* to evaluate the recommended sets.

4.4 Empirical study

We conduct an empirical study to investigate the correlation of items in RS and their corresponding entities in KG. Specifically, we aim to reveal how the number of common neighbors of an item pair in KG changes with their number of common raters in RS. To this end, we first randomly sample 1 million item pairs from MovieLens-1M. We then classify each pair into 5 categories based on the number of their common raters in RS, and count their average number of common neighbors in KG for each category. The result is presented in Figure 2a, which clearly shows that *if two items have more common raters in RS, they are likely to share more common neighbors in KG*. Figure 2b shows the positive correlation from an opposite direction. The above findings empirically demonstrate that *items share the similar structure of proximity in KG and RS*, thus the cross knowledge transfer of items benefits both recommendation and KGE tasks in MKR.

4.5 Results

4.5.1 Comparison with baselines. The results of all methods in CTR prediction and top-K recommendation are presented in Table 2 and Figure 3, 4, respectively. We have the following observations:

- PER performs poor on movie, book, and music recommendation because the user-defined meta-paths can hardly be optimal in reality. Moreover, PER cannot be applied to news recommendation.
- CKE performs better in movie, book, and music recommendation than news. This may be because MovieLens-1M, Book-Crossing, and Last.FM are much denser than Bing-News, which is more favorable for the collaborative filtering part in CKE.
- DKN performs best in news recommendation compared with other baselines, but performs worst in other scenarios. This is because movie, book, and musician names are too short and ambiguous to provide useful information.
- RippleNet performs best among all baselines, and even outperforms MKR on MovieLens-1M. This demonstrates that RippleNet can precisely capture user interests, especially in the case where user-item interactions are dense. However, RippleNet is more sensitive to the density of datasets, as it performs worse than MKR in Book-Crossing, Last.FM, and Bing-News. We will further study their performance in sparse scenarios in Section 4.5.3.
- In general, our MKR performs best among all methods on the four datasets. Specifically, MKR achieves average *Accuracy* gains of 11.6%, 11.5%, 12.7%, and 8.7% in movie, book, music, and news recommendation, respectively, which demonstrates the efficacy of the multi-task learning framework in MKR. Note that the top-K metrics are much lower for Bing-News because the number of news is significantly larger than movies, books, and musicians.

4.5.2 Comparison with MKR variants. We further compare MKR with its three variants to demonstrate the efficacy of cross&compress unit:

- MKR-1L is MKR with one layer of cross&compress unit, which corresponds to FM model according to Proposition 1. Note that MKR-1L is actually MKR in the experiments for MovieLens-1M.
- MKR-DCN is a variant of MKR based on Eq. (13), which corresponds to DCN model.
- MKR-stitch is another variant of MKR corresponding to the cross-stitch network, in which the transfer weights in Eq. (15) are replaced by four trainable scalars.

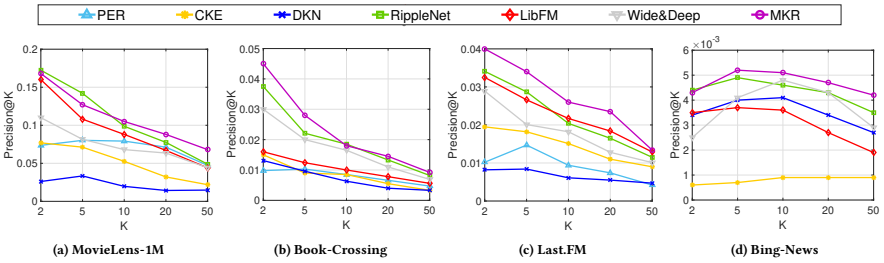


Figure 3: The results of *Precision@K* in top-*K* recommendation.

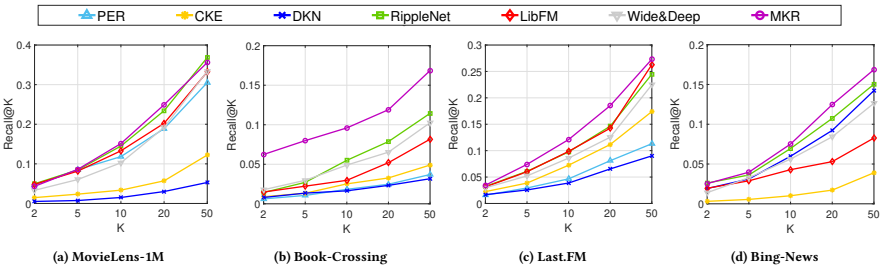


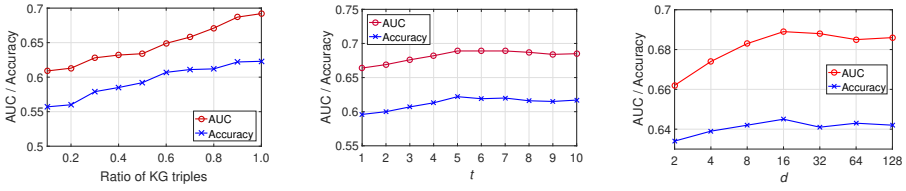
Figure 4: The results of *Recall@K* in top-*K* recommendation.

Table 3: Results of *AUC* on MovieLens-1M in CTR prediction with different ratios of training set *r*.

Model	*r*									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
PER	0.598	0.607	0.621	0.638	0.647	0.662	0.675	0.688	0.697	0.710
CKE	0.674	0.692	0.705	0.716	0.739	0.754	0.768	0.775	0.797	0.801
DKN	0.579	0.582	0.589	0.601	0.612	0.620	0.631	0.638	0.646	0.655
RippleNet	0.843	0.851	0.859	0.862	0.870	0.878	0.890	0.901	0.912	**0.920**
LibFM	0.801	0.810	0.816	0.829	0.837	0.850	0.864	0.875	0.886	0.892
Wide&Deep	0.788	0.802	0.809	0.815	0.821	0.840	0.858	0.876	0.884	0.898
MKR	**0.868**	**0.874**	**0.881**	**0.882**	**0.889**	**0.897**	**0.903**	**0.908**	**0.913**	0.917

From Table 2 we observe that MKR outperforms MKR-1L and MKR-DCN, which shows that modeling high-order interactions between item and entity features is helpful for maintaining decent performance. MKR also achieves better scores than MKR-stitch. This validates the efficacy of fine-grained control on knowledge transfer in MKR compared with the simple cross-stitch units.

4.5.3 Results in sparse scenarios. One major goal of using knowledge graph in MKR is to alleviate the sparsity and the cold start problem of recommender systems. To investigate the efficacy of the KGE module in sparse scenarios, we vary the ratio of training set of MovieLens-1M from 100% to 10% (while the validation and test set are kept fixed), and report the results of *AUC* in CTR prediction for all methods. The results are shown in Table 3. We observe that the performance of all methods deteriorates with the reduce of the training set. When $r = 10\%$, the *AUC* score decreases by 15.8%, 15.9%, 11.6%, 8.4%, 10.2%, 12.2% for PER, CKE, DKN, RippleNet, LibFM, and Wide&Deep, respectively, compared with the case when full training set is used ($r = 100\%$). In contrast, the



(a) KG size (b) Training frequency (c) Dimension of embeddings

Figure 5: Parameter sensitivity of MKR on Bing-News w.r.t. (a) the size of the knowledge graph; (b) training frequency of the RS module t; and (c) dimension of embeddings d.

Table 4: The results of *RMSE* on the KGE module for the three datasets. "KGE" means only KGE module is trained, while "KGE + RS" means KGE module and RS module are trained together.

dataset	KGE	KGE + RS
MovieLens-1M	0.319	**0.302**
Book-Crossing	0.596	**0.558**
Last.FM	0.480	**0.471**
Bing-News	0.488	**0.459**

AUC score of MKR only decreases by 5.3%, which demonstrates that MKR can still maintain a decent performance even when the user-item interaction is sparse. We also notice that MKR performs better than RippleNet in sparse scenarios, which is accordance with our observation in Section 4.5.1 that RippleNet is more sensitive to the density of user-item interactions.

4.5.4 Results on KGE side. Although the goal of MKR is to utilize KG to assist with recommendation, it is still interesting to investigate whether the RS task benefits the KGE task, since the principle of multi-task learning is to leverage shared information to help improve the performance of all tasks [42]. We present the result of *RMSE* (rooted mean square error) between predicted and real vectors of tails in the KGE task in Table 4. Fortunately, we find that the existence of RS module can indeed reduce the prediction error by 1.9% ∼ 6.4%. The results show that the cross&compress units are able to learn general and shared features that mutually benefit both sides of MKR.

4.6 Parameter Sensitivity

4.6.1 Impact of KG size. We vary the size of KG to further investigate the efficacy of usage of KG. The results of *AUC* on Bing-News are plotted in Figure 5a. Specifically, the *AUC* and *Accuracy* is enhanced by 13.6% and 11.8% with the KG ratio increasing from 0.1 to 1.0 in three scenarios, respectively. This is because the Bing-News dataset is extremely sparse, making the effect of KG usage rather obvious.

4.6.2 Impact of RS training frequency. We investigate the influence of parameters t in MKR by varying t from 1 to 10, while keeping other parameters fixed. The results are presented in Figure 5b. We observe that MKR achieves the best performance when $t = 5$. This is because a high training frequency of the KGE module will mislead the objective function of MKR, while too small of a training frequency of KGE cannot make full use of the transferred knowledge from the KG.

4.6.3 Impact of embedding dimension. We also show how the dimension of users, items, and entities affects the performance of MKR in Figure 5c. We find that the performance is initially improved with the increase of dimension, because more bits in embedding layer can encode more useful information. However, the performance drops when the dimension further increases, as too large number of dimensions may introduce noises which mislead the subsequent prediction.

5 RELATED WORK

5.1 Knowledge Graph Embedding

The KGE module in MKR connects to a large body of work in KGE methods. KGE is used to embed entities and relations in a knowledge into low-dimensional vector spaces while still preserving the structural information [33]. KGE methods can be classified into the following two categories: (1) Translational distance models exploit distance-based scoring functions when learning representations of entities and relations, such as TransE [2], TransH [35], and TransR [13]; (2) Semantic matching models measure plausibility of knowledge triples by matching latent semantics of entities and relations, such as RESCAL [20], ANALOGY [19], and HolE [14]. Recently, researchers also propose incorporating auxiliary information, such as entity types [36], logic rules [24], and textual descriptions [46] to assist KGE. The above KGE methods can also be incorporated into MKR as the implementation of the KGE module, but note that the cross&compress unit in MKR needs to be redesigned accordingly. Exploring other designs of KGE module as well as the corresponding bridging unit is also an important direction of future work.

5.2 Multi-Task Learning

Multi-task learning is a learning paradigm in machine learning and its aim is to leverage useful information contained in multiple related tasks to help improve the generalization performance of all the tasks [42]. All of the learning tasks are assumed to be related to

each other, and it is found that learning these tasks jointly can lead to performance improvement compared with learning them individually. In general, MTL algorithms can be classified into several categories, including feature learning approach [34, 41], low-rank approach [7, 16], task clustering approach [47], task relation learning approach [12], and decomposition approach [6]. For example, the cross-stitch network [41] determines the inputs of hidden layers in different tasks by a knowledge transfer matrix; Zhou et. al [47] aims to cluster tasks by identifying representative tasks which are a subset of the given m tasks, i.e., if task T_i is selected by task T_j as a representative task, then it is expected that model parameters for T_j are similar to those of T_i. MTL can also be combined with other learning paradigms to improve the performance of learning tasks further, including semi-supervised learning, active learning, unsupervised learning,and reinforcement learning.

Our work can be seen as an asymmetric multi-task learning framework [37, 43, 44], in which we aim to utilize the connection between RS and KG to help improve their performance, and the two tasks are trained with different frequencies.

5.3 Deep Recommender Systems

Recently, deep learning has been revolutionizing recommender systems and achieves better performance in many recommendation scenarios. Roughly speaking, deep recommender systems can be classified into two categories: (1) Using deep neural networks to process the raw features of users or items [5, 28–30, 40]; For example, Collaborative Deep Learning [29] designs autoencoders to extract short and dense features from textual input and feeds the features into a collaborative filtering module; DeepFM [5] combines factorization machines for recommendation and deep learning for feature learning in a neural network architecture. (2) Using deep neural networks to model the interaction among users and items [3, 4, 8, 9]. For example, Neural Collaborative Filtering [8] replaces the inner product with a neural architecture to model the user-item interaction. The major difference between these methods and ours is that MKR deploys a multi-task learning framework that utilizes the knowledge from a KG to assist recommendation.

6 CONCLUSIONS AND FUTURE WORK

This paper proposes MKR, a multi-task learning approach for knowledge graph enhanced recommendation. MKR is a deep and end-to-end framework that consists of two parts: the recommendation module and the KGE module. Both modules adopt multiple nonlinear layers to extract latent features from inputs and fit the complicated interactions of user-item and head-relation pairs. Since the two tasks are not independent but connected by items and entities, we design a cross&compress unit in MKR to associate the two tasks, which can automatically learn high-order interactions of item and entity features and transfer knowledge between the two tasks. We conduct extensive experiments in four recommendation scenarios. The results demonstrate the significant superiority of MKR over strong baselines and the efficacy of the usage of KG.

For future work, we plan to investigate other types of neural networks (such as CNN) in MKR framework. We will also incorporate other KGE methods as the implementation of KGE module in MKR by redesigning the cross&compress unit.

APPENDIX

A Proof of Theorem 1

Proof. We prove the theorem by induction:

Base case: When $l = 1$,

$$\begin{aligned}\mathbf{v}_1 =& \mathbf{v}\mathbf{e}^\top \mathbf{w}_0^{VV} + \mathbf{e}\mathbf{v}^\top \mathbf{w}_0^{EV} + \mathbf{b}_0^V \\ =& \left[v_1 \sum_{i=1}^{d} e_i w_0^{VV(i)} \ \cdots \ v_d \sum_{i=1}^{d} e_i w_0^{VV(i)}\right]^\top \\ &+ \left[e_1 \sum_{i=1}^{d} v_i w_0^{EV(i)} \ \cdots \ e_d \sum_{i=1}^{d} v_i w_0^{EV(i)}\right]^\top \\ &+ \left[b_0^{V(0)} \ \cdots \ b_0^{V(d)}\right]^\top.\end{aligned}$$

Therefore, we have

$$\begin{aligned}\|\mathbf{v}_1\|_1 =& \left|\sum_{j=1}^{d} v_j \sum_{i=1}^{d} e_i w_0^{VV(i)} + \sum_{j=1}^{d} e_j \sum_{i=1}^{d} v_i w_0^{EV(i)} + \sum_{i=1}^{d} b_0^{V(d)}\right| \\ =& \left|\sum_{i=1}^{d}\sum_{j=1}^{d} (w_0^{EV(i)} + w_0^{VV(j)}) v_i e_j + \sum_{i=1}^{d} b_0^{V(d)}\right|.\end{aligned}$$

It is clear that the cross terms about $\mathbf{v}$ and $\mathbf{e}$ with maximal degree is $k_{\boldsymbol{\alpha},\boldsymbol{\beta}} v_i e_j$, so we have $\alpha_1 + \cdots + \alpha_d = 1 = 2^{1-1}$, and $\beta_1 + \cdots + \beta_d = 1 = 2^{1-1}$ for $\mathbf{v}_1$. The proof for $\mathbf{e}_1$ is similar.

Induction step: Suppose $\alpha_1 + \cdots + \alpha_d = 2^{l-1}$ and $\beta_1 + \cdots + \beta_d = 2^{l-1}$ hold for the maximal-degree term x and y in $\|\mathbf{v}_l\|_1$ and $\|\mathbf{e}_l\|_1$. Since $\|\mathbf{v}_l\|_1 = \left|\sum_{i=1}^{d} v_l^{(i)}\right|$ and $\|\mathbf{e}_l\|_1 = \left|\sum_{i=1}^{d} e_l^{(i)}\right|$, without loss of generosity, we assume that x and y exist in $v_l^{(a)}$ and $e_l^{(b)}$, respectively. Then for $l+1$, we have

$$\|\mathbf{v}_{l+1}\|_1 = \sum_{i=1}^{d}\sum_{j=1}^{d} (w_l^{EV(i)} + w_l^{VV(j)}) v_l^{(i)} e_l^{(j)} + \sum_{i=1}^{d} b_l^{V(d)}.$$

Obviously, the maximal-degree term in $\|\mathbf{v}_{l+1}\|_1$ is the cross term xy in $v_l^{(a)} e_l^{(b)}$. Since we have $\alpha_1 + \cdots + \alpha_d = 2^{l-1}$ and $\beta_1 + \cdots + \beta_d = 2^{l-1}$ for both x and y, the degree of cross term xy therefore satisfies $\alpha_1 + \cdots + \alpha_d = 2^{(l+1)-1}$ and $\beta_1 + \cdots + \beta_d = 2^{(l+1)-1}$. The proof for $\|\mathbf{e}_{l+1}\|_1$ is similar. □

B Proof of Proposition 1

Proof. In the proof of Theorem 1 in Appendix A, we have shown that

$$\|\mathbf{v}_1\|_1 = \left|\sum_{i=1}^{d}\sum_{j=1}^{d} (w_0^{EV(i)} + w_0^{VV(j)}) v_i e_j + \sum_{i=1}^{d} b_0^{V(d)}\right|.$$

It is easy to see that $w_i = w_0^{EV(i)}$, $w_j = w_0^{VV(j)}$, and $b = \sum_{i=1}^{d} b_0^{V(d)}$. The proof is similar for $\|\mathbf{e}_1\|_1$. □

We omit the proofs for Proposition 2 and Proposition 3 as they are straightforward.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*. 2787–2795.

[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.

[5] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*.

[6] Lei Han and Yu Zhang. 2015. Learning tree structure in multi-task learning. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 397–406.

[7] Lei Han and Yu Zhang. 2016. Multi-Stage Multi-Task Learning with Reduced Rank.. In *AAAI*. 1638–1644.

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. 173–182.

[9] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*. ACM, 2333–2338.

[10] Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the 4th ACM conference on Recommender systems*. ACM, 135–142.

[11] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[12] Giwoong Lee, Eunho Yang, and Sung Hwang. 2016. Asymmetric multi-task learning based on task relatedness and loss. In *International Conference on Machine Learning*. 230–238.

[13] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion.. In *The 29th AAAI Conference on Artificial Intelligence*. 2181–2187.

[14] Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical Inference for Multi-Relational Embeddings. In *Proceedings of the 34th International Conference on Machine Learning*. 2168–2178.

[15] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and S Yu Philip. 2017. Learning Multiple Tasks with Multilinear Relationship Networks. In *Advances in Neural Information Processing Systems*. 1593–1602.

[16] Andrew M McDonald, Massimiliano Pontil, and Dimitris Stamos. 2014. Spectral k-support norm regularization. In *Advances in Neural Information Processing Systems*. 3644–3652.

[17] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. 3111–3119.

[18] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3994–4003.

[19] Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. 2016. Holographic Embeddings of Knowledge Graphs.. In *The 30th AAAI Conference on Artificial Intelligence*. 1955–1961.

[20] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data. In *Proceedings of the 28th International Conference on Machine Learning*. 809–816.

[21] Sinno Jialin Pan, Qiang Yang, et al. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2010), 1345–1359.

[22] Steffen Rendle. 2010. Factorization machines. In *Proceedings of the 10th IEEE International Conference on Data Mining*. IEEE, 995–1000.

[23] Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 3 (2012), 57.

[24] Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1119–1129.

[25] Walter Rudin et al. 1964. *Principles of mathematical analysis*. Vol. 3. McGraw-hill New York.

[26] Jie Tang, Sen Wu, Jimeng Sun, and Hang Su. 2012. Cross-domain collaboration recommendation. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1285–1293.

[27] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. Graphgan: Graph representation learning with generative adversarial nets. In *AAAI*. 2508–2515.

[28] Hongwei Wang, Jia Wang, Miao Zhao, Jiannong Cao, and Minyi Guo. 2017. Joint Topic-Semantic-aware Social Recommendation for Online Voting. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 347–356.

[29] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1235–1244.

[30] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. 2018. Shine: Signed heterogeneous information network embedding for sentiment link prediction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 592–600.

[31] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM.

[32] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep Knowledge-Aware Network for News Recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1835–1844.

[33] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[34] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the ADKDD'17*. ACM, 12.

[35] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1591–1601.

[36] Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Hierarchical Types.. In *IJCAI*. 2965–2971.

[37] Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. 2007. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research* 8, Jan (2007), 35–63.

[38] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. In *Advances in Neural Information Processing Systems*. 3320–3328.

[39] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*. 283–292.

[40] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 353–362.

[41] Wenlu Zhang, Rongjian Li, Tao Zeng, Qian Sun, Sudhir Kumar, Jieping Ye, and Shuiwang Ji. 2015. Deep model based transfer and multi-task learning for biological image analysis. In *21st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2015*. Association for Computing Machinery.

[42] Yu Zhang and Qiang Yang. 2017. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114* (2017).

[43] Yu Zhang and Dit-Yan Yeung. 2012. A convex formulation for learning task relationships in multi-task learning. *arXiv preprint arXiv:1203.3536* (2012).

[44] Yu Zhang and Dit-Yan Yeung. 2014. A regularization approach to learning task relationships in multitask learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 8, 3 (2014), 12.

[45] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-graph based recommendation fusion over heterogeneous information networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 635–644.

[46] Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. 2015. Aligning knowledge and text embeddings by entity descriptions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 267–272.

[47] Qiang Zhou and Qi Zhao. 2016. Flexible Clustered Multi-Task Learning by Learning Representative Tasks. *IEEE Trans. Pattern Anal. Mach. Intell.* 38, 2 (2016), 266–278.

The 15th International Conference on Document Analysis and Recognition

ICDAR 2019 Robust Reading Challenge on Reading Chinese Text on Signboard

Xi Liu[1], Rui Zhang[1], Yongsheng Zhou[1], Qianyi Jiang[1], Qi Song[1], Nan Li[1], Kai Zhou[1], Lei Wang[1], Dong Wang[1], Minghui Liao[2], Mingkun Yang[2], Xiang Bai[2], Baoguang Shi[3], Dimosthenis Karatzas[4], Shijian Lu[5]

[1]Meituan-Dianping Group, China, [2]School of EIC, Huazhong University of Science and Technology, China,

[3]Microsoft Redmond, USA, [4]Computer Vision Centre, UAB, Spain, [5]Nanyang Technological University, Singapore

***Abstract*—Chinese scene text reading is one of the most challenging problems in computer vision and has attracted great interest. Different from English text, Chinese has more than 6000 commonly used characters and Chinese characters can be arranged in various layouts with numerous fonts. The Chinese signboards in street view are a good choice for Chinese scene text images since they have different backgrounds, fonts and layouts. We organized a competition called ICDAR2019-ReCTS, which mainly focuses on reading Chinese text on signboard. This report presents the final results of the competition. A large-scale dataset of 25,000 annotated signboard images, in which all the text lines and characters are annotated with locations and transcriptions, were released. Four tasks, namely character recognition, text line recognition, text line detection and end-to-end recognition were set up. Besides, considering the Chinese text ambiguity issue, we proposed a multi ground truth (multi-GT) evaluation method to make evaluation fairer. The competition started on March 1, 2019 and ended on April 30, 2019. 262 submissions from 46 teams are received. Most of the participants come from universities, research institutes, and tech companies in China. There are also some participants from the United States, Australia, Singapore, and Korea. 21 teams submit results for Task 1, 23 teams submit results for Task 2, 24 teams submit results for Task 3, and 13 teams submit results for Task 4. The official website for the competition is http://rrc.cvc.uab.es/?ch=12.**

I. Introduction

Texts in natural images carry much important semantic information. Reading text in natural scene images has been widely studied recently since it is an important prerequisite for many content-based image analysis tasks such as photo translation, fine-grained image classification and autonomous driving.

It is widely recognized that large-scale, well-annotated datasets are crucial to today's deep learning based techniques. In scene text reading field, many scene text datasets have been collected. Especially for Chinese text reading, more and more Chinese scene text datasets are proposed, such as MSRA-500 [1], RCTW [2], SCUT-CTW1500 [3], CTW [4].

Chinese text reading is a huge challenge task. Different from English text reading, Chinese has more than 6000 commonly used characters. Besides, owing to the Chinese culture, the layouts, arrangements and fonts of Chinese characters are always in a great variety, as shown in Figure 1.

The Chinese signboards in street view may be the best source for Chinese scene text images since they have different backgrounds, fonts and layouts. In Meituan-Dianping Group, a Chinese leading company for food delivery services, consumer products and retail services, there are many signboard images collected by Meituan business merchants. Based on this, we propose a competition for Chinese text reading on signboard and construct a large-scale challenging natural scene text dataset of 25,000 signboard images. About 200,000 text lines and 600,000 characters are labeled with locations and transcriptions. We set up four tasks for this competition, namely character recognition, text line recognition, text line detection and end-to-end recognition. Besides, we propose a multi ground truth (multi-GT) evaluation method considering the Chinese text ambiguity. As illustrated in Figure 2, it is difficult to determine whether some words should be merged to a text instance or not. We thus provide one or more ground truths for each test image and compare the predicted result with all the ground truths when evaluating. The best matched GT will be used to calculate the evaluation metrics.

Figure 1. Characters with various layouts and fonts.

Figure 2. Chinese text ambiguity in signboard image.

Figure 3. Chinese character test images.

Figure 4. Text line test images.

The competition lasts from March 1st to April 30, 2019. It receives lots of attention from the community. For all the four tasks, there are all together 46 valid teams participating in the competition and hundreds of valid submissions are received. In this report, we will present their evaluation results.

II. DATASET AND ANNOTATIONS

The dataset, named ReCTS-25k, comprises 25,000 signboard images. All the images are from Meituan-Dianping Group, collected by Meituan business merchants, using phone cameras under uncontrolled conditions. Different from other datasets, this dataset mainly focuses on Chinese text reading on the signboards. The layout and arrangement of Chinese characters in signboards are much more complex for the sake of aesthetics appearance or highlighting certain elements. Figure 1 shows some example images.

We manually annotate the locations and transcriptions for all the text lines and characters in the signboard images. Note that the utterly obscure and small text lines and characters are marked with a difficult flag. Locations are annotated in terms of polygons with four vertices, which are in clockwise order starting from the upper left vertice. Transcriptions are UTF-8 encoded strings.

The dataset is split into two subsets. The training set consists of 20,000 images, and the test set consists of 5,000 images. Moreover, 29335 character images and 10789 text lines images, cropped from the 5000 test images, are used for task 1 and task 2 evaluation respectively.

III. CHALLENGE TASKS

Robust reading challenge on Chinese signboard consists of four tasks: 1) Character recognition, 2) Text line recognition, 3) Text line detection, 4) End-to-end recognition. Given that Chinese signboards have various layouts, fonts and orientations, character and text line reading are concerned. Therefore, in our competition, character based and text line based tasks are both evaluated.

Note that the half-width character and its corresponding full-width character are regarded as one character in the evaluation of task 2 and task 4. Moreover, the English letters are not case sensitive.

A. Task 1 – Character Recognition

The aim of this task is to recognize characters of the cropped character images from Chinese signboards. As illustrated in Figure 3, the Chinese characters take the largest portion and are in diverse fonts. Participant is asked to submit a text file containing character results for all test images. The recognition accuracy is given as the metric:

$$\text{accuracy} = \frac{\mathrm{N_{right}}}{N_{total}}, \tag{1}$$

where $\mathrm{N_{right}}$ is the number of characters predicted correctly and $\mathrm{N_{total}}$ is the total number of the test characters.

B. Task 2 – Text Line Recognition

The target of text line recognition is to recognize the cropped word images of scene text. The cropped text line images as well as the coordinates of the polygon bounding boxes in the images are given. The given points are arranged in the clockwise order, starting from the top-left point. Figure 4 shows some examples of the test set. The text line images may contain perspective and arbitrary arranged text lines.

The results are evaluated by the Normalized Edit Distance between the recognition result and the ground truth. The edit distances are summarized and divided by the number of test images. The resulting average edit distance is taken as the metric and is formulated as follows:

$$\text{accuracy} = 1 - \frac{1}{N}\sum_{i=1}^{N}\frac{D(s_i,\hat{s_i})}{\max(s_i,\hat{s_i})}, \tag{2}$$

where D stands for the Levenshtein Distance, s_i denotes the predicted text line and $\hat{s_i}$ denotes the corresponding ground truth, N is the total number of text lines.

C. Task 3 – Text Line Detection

The aim of this task is to localize text lines in the signboard. The input image is the full signboard images. The detection results submitted by the participants are required to give four vertices of the polygon in clockwise order.

In some signboard, there always exist the following case, as shown in Figure 2. It is difficult to determine whether the boxes "砂锅" , "炒面" , "拌面" , "烩肉" , "泡馍" should be merged to a large text box or not. Therefore, we regard the two cases (Figure 2(a) and Figure 2(b)) as correct ground truth. We provide one or more ground truths for each test image. When

evaluating, we compare the predicted result with all the ground truths and use the best matched one to calculate the evaluation metrics.

Following the evaluation protocols of ICDAR 2017-RCTW [2] dataset, the detection task is evaluated in terms of Precision, Recall and F-score with intersection-over-union (IoU) threshold of 0.5 and 0.7. The F-score at IoU=0.5 will be used as the only metric for the final ranking. All detected or missed ignored ground truths will not contribute to the evaluation result.

D. Task 4 – End-to-End Recognition

The aim of this task is to localize and recognize every text instance in the signboard. The input image is the full signboard images. Participants are required to submit the text file containing all the recognized text lines locations and transcriptions for each test image. Similar to Task 3, the locations are four vertices in clock-wise order and the transcripts are UTF-8 encoded strings.

The evaluation process consists of two steps. First, each detection is matched to a ground truth polygon that has the maximum IOU, or it is matched to 'None' if none IOU is larger than 0.5. If multiple detections are matched to the same ground-truth, only the one with the maximum IOU will be kept and the others are recorded as 'None'. Then, we calculate the edit distances between all matching pairs by Formula (2). Since one test image may have multiple ground truths, as stated in Task 3, we also compare the predicted result with all the ground truths and use the best matched one to calculate the evaluation metrics.

IV. ORGANIZATION

The competition starts on March 1, 2019, when the RRC website is ready and open for registration. The training set is released on March 18, the first part of test set is released on April 12 and the second part of test set released on April 20. We revise the test set more than once to fixed some errors before releasing the test set. The RRC website opens for result submission on April 20 and closes at 11:59 PM PST, April 30.

There are all together 46 valid teams participated in the competition. Most of the participants come from universities, research institutes, and tech companies in China. There are also some participants from the United States, Australia, Singapore, and Korea.

All the teams submit their results through the RRC website. Each team is allowed to submit 5 results at most and we choose the best result among the 5 results as the final result. 21 teams submit results for Task 1, 23 teams submit results for Task 2, 24 teams submit results for Task 3, and 13 teams submit results for Task 4.

V. SUBMISSIONS AND RESULTS

The evaluation script is implemented in Python. We run the script to evaluate all the submissions. Table I summarizes the top 5 results of Task 1. Methods are ranked by their accuracy. Table II summarizes the top 5 results of Task 2. Methods are ranked by their normalized edit distance. Table III summarizes the top 5 results of Task 3. Methods are ranked by their F-score. Table IV summarizes the top 5 results of Task 4. Methods are ranked by their normalized edit distance. You can view the complete ranking in the home page of the competition https://rrc.cvc.uab.es/?ch=12.

A. Top 3 submissions for Task 1

1. "BASELINE v1" (USTC-iFLYTEK) The method uses image classification methods and its ensemble.

2. "Amap_CVLab" (Alibaba AMAP) The method adds res-block [5] (for the lower dimension feature collapse avoiding) and se-block [6]. Their training dataset contains both the ReCTS-25k and other data.

3. "TPS-ResNet v1" (Clova AI OCR Team, NAVER/LINE Corp) The method uses Thin-plate-spline(TPS) [7] based Spatial transformer network (STN) [8], which normalizes the input text images. They use ResNet [5], BiLSTM [9] and attention mechanism. Their training dataset contains the Chinese synthetic datasets (MJSynth and SynthText [10]) and real dataset (ArT [11], LSVT [12], RCTW [2], ReCTS-25k).

B. Top 3 submissions for Task 2

1. "SANHL" (South China University of Technology, Northwestern Polytechnical University, The University of Adelaide, Lenovo and Huawei) The method uses an ensemble framework, which consists of attention-based network, transformer network and CTC-based [13] network. Apart from the official training dataset, about 2 million synthesized samples are used for training.

2. "Tencent-DPPR Team" (Tencent-DPPR Team) The method uses five types of deep models, which mainly include CTC-based nets and multi-head attention based nets. All samples are resized to the same height before feeding into the network. Furthermore, besides ReCTS, they use a synthetic dataset containing more than fifty million images, as well as open-source datasets including LSVT [12], COCO-Text [14], RCTW [2] and ICPR-2018-MTWI. In terms of data augmentation, they mainly use Gaussian blur, Gaussian noise and so on.

3. "HUST_VLRGROUP" (Huazhong University of Science and Technology) A CRNN based method.

C. Top 3 submissions for Task 3

1. "SANHL_v4" (South China University of Technology, The University of Adelaide, Northwestern Polytechnical University, Lenovo, HUAWEI) The method uses a sequential-free box discretization method to localize the text instances. Multi-scale testing and model ensemble are used to generate the final result. Their training dataset contains LSVT [12], ArT [11], MLT [15] and ReCTS-25k.

2. "Tencent-DPPR Team" (Tencent Data Platform Precision Recommendation) Their text detector is based on two-stage method with multi-scale training policy, and ResNet101 [5] is used as the backbone network. They use feature pyramid layers [16] to extract features instead of choosing one layer according to box sizes. They use LSVT [12] pre-trained model.

3. "Amap-CVLab" (Alibaba AMAP, Alibaba DAMO Academy for Discovery, Adventure, Momentum and Outlook) The method is based on Mask R-CNN [17]. Their training dataset contains RCTW[2], ICDAR2017-MLT[15], LSVT[12], ReCTS-25k.

D. Top 3 submissions for Task 4

1. "Tencent-DPPR Team" (Tencent-DPPR Team) In the detection part, they use a text detector based on two-stage method. This method uses ResNet101 [5] as feature extractor, and they design a policy to help proposals select feature pyramid layers [16] to extract features instead of choosing one layer according to box sizes. In detection ensemble stage, they apply a multi-scale test method with different backbones. When ensembling all the results, they develop an approach to vote boxes after scoring each box. In the recognition part, they use an ensemble model, which includes CTC-based nets and multi-head attention based nets. For this task, they use the predicted confidence scores of cropped words and the ensemble results to select the reliable one among results predicted by all models.

2. "SANHL" (South China University of Technology, Northwestern Polytechnical University, The University of Adelaide, Lenovo and Huawei) The method firstly detect possible text lines, and then predict strings by an ensembled recognition model.

3. "HUST_VLRGROUP" (Huazhong University of Science and Technology) The method uses Mask R-CNN as text detector and a CRNN based approach to predict strings.

E. Baseline submissions

For reference, we submit a baseline method to Task 1, Task 2, Task 3 and Task 4 respectively. The methods are implemented by ourselves. Their results are shown in Table I, II, III and IV.

For Task 1, the character Recognition method is based on the densely connected convolutional network (DenseNet) [18]. Our network inherits from the DenseNet-169 network model with dense blocks, but we reduce the number of last dense block to 24 and all the growth rates in the networks are 32. The training dataset consists of ReCTS and synthetic data.

For Task 2, We took the Chinese text line recognition as a sequence recognition task. We utilized a modified version of Inception-V4 [19], integrated with attention module to extract feature maps. The CTC layer for transcription is adopted. The baseline result is obtained by a single recognition model, the training dataset consists of ReCTS, RCTW [2], and LSVT [12], no synthetic data is utilized.

For Task 3, the text detection method is based on SEG-FPN [20] and Pixel-link [21]. We build a unified framework, which combines pixel link and segment link in feature pyramid network to detect scene text. The training dataset only consists of ReCTS.

For Task 4, we first detect the text line in the image. If the text line is horizontal, recognize it by the line recognition model; if the text line is vertical, character detection and character recognition model will be used. The text line detection part is the same as that for Task 3, the character recognition part is the same as that for Task 1, and the text line recognition part is the same as that for Task 2. A Faster-RCNN [22] based detection approach is adopted to detect Chinese character regions.

VI. CONCLUSIONS

We organize the competition on reading Chinese text on signboard (ReCTS). A large-scale challenging natural scene text dataset of 25,000 signboard images are released and four tasks are set up. We also propose a multi-GT evaluation strategy intended for Chinese text ambiguity. During the challenge, we receive hundreds of submissions from 46 teams, which shows the broad interest in the community. In the future, we plan to make the evaluation scripts available on the website https://rrc.cvc.uab.es/ and users can get the evaluation results shortly after they submit the results to the website.

REFERENCES

1. C. Yao, X. Bai, W. Liu, Y. Ma, Z. Tu, "Detecting Texts of Arbitrary Orientations in Natural Images," CVPR, 2012.
2. B. Shi, C. Yao, M. Liao, M. Yang, P. Xu, L. Cui, S. J. Belongie, S. Lu, X. Bai, "ICDAR2017 Competition on Reading Chinese Text in the Wild (RCTW-17)," ICDAR, 2017.
3. Y. L. Liu, L. W. Jin, S. T. Zhang, S. Zhang, "Detecting Curve Text in the Wild: New Dataset and New Solution," arXiv, 2017.
4. T. L. Yuan, Z. Zhu, K. Xu, "Chinese Text in the Wild," arXiv, 2018.
5. K. He, X. Zhang, S. Ren, "Deep Residual Learning for Image Recognition," CVPR, 2016.
6. J. Hu, L. Shen, G. Sun, "Squeeze-and-Excitation Networks," TPAMI, 2017.
7. F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," TPAMI, 1989.
8. M. Jaderberg, K. Simonyan, A. Zisserman, K. Kavukcuoglu, "Spatial Transformer Networks," CVPR, 2016.
9. A. Graves, J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," Neural Networks, 2005.
10. A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," In CVPR, pages 2315–2324, 2016.
11. https://rrc.cvc.uab.es/?ch=14, 2019.
12. https://rrc.cvc.uab.es/?ch=16, 2019.
13. A. Graves, S. Fernandez, F. Gomez, J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," ICML, 2006.
14. V. Andreas, M. Tomas, N. Lukas, M. Jiri, B. Serge, "COCO-Text: Dataset and Benchmark for Text Detection and Recognition in Natural Images," arXiv, 2016.
15. N. Nayef, et al, "ICDAR2017 Robust Reading Challenge on Multi-lingual Scene Text Detection and Script Identification," ICDAR 2017.
16. Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie, "Feature Pyramid Networks for Object Detection," arXiv preprint. arXiv: 1612.03144, 2017.
17. K. He, G. Gkioxari, P. Dollar, R. Girshick, "Mask r-cnn," ICCV, 2017.
18. G. Hunag, Z. Liu, L. V. DerMaaten, K. Q. Weinberger, "Densely connected convolutional networks." CVPR, 2017.
19. C. Szegedy, et al, "Inception-V4, inception-resnet and the impact of residual connections on learning," AAAI, 2017.
20. X. Liu, R. Zhang, Y. S. Zhou, D. Wang, "Scene Text Detection with Feature Pyramid Network and Linking Segments," ICDAR, 2019.
21. D. Deng, H. Liu, X. Li, and D. Cai, "Pixellink: Detecting scene text via instance segmentation," In AAAI, pages 6773– 6780, 2018.
22. S. Q. Ren, K. He, R. Girshick, J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," NIPS, 2015.

TABLE I: RESULTS SUMMARY FOR THE TOP-5 SUBMISSIONS OF TASK 1.

Ranking	Team Name	Affiliation	Accuracy
1	BASELINE-v1	iFLYTEK, University of Science and Technology of China	0.9737
2	Amap_CVLab	Alibaba AMAP	0.9728
3	TPS-ResNet-v1	Clova AI OCR Team, NAVER/LINE Corp	0.9612
4	SANHL_v4	South China University of Technology, The University of Adelaide, Northwestern Polytechnical University, Lenovo, HUAWEI	0.9594
5	Tencent-DPPR	Tencent (Data Platform Precision Recommendation)	0.9512
Baseline		**Meituan Dianping**	**0.9140**

TABLE II: RESULTS SUMMARY FOR THE TOP-5 SUBMISSIONS OF TASK 2.

Ranking	Team Name	Affiliation	N.E.D
1	SANHL_v1	South China University of Technology, The University of Adelaide, Northwestern Polytechnical University, Lenovo, HUAWEI	0.9555
2	Tencent-DPPR	Tencent (Data Platform Precision Recommendation)	0.9486
3	HH-Lab-v4 *	Huazhong University of Science and Technology (Visual and Learning Representation Group)	0.9483
4	TPS-ResNet-v1	Clova AI OCR Team, NAVER/LINE Corp	0.9477
5	Baseline-Beihang*	Beihang University	0.9437
Baseline		**Meituan Dianping**	**0.9089**

TABLE III: RESULTS SUMMARY FOR THE TOP-5 SUBMISSIONS OF TASK 3.

Ranking	Team Name	Affiliation	F-score
1	SANHL_v4	South China University of Technology, The University of Adelaide, Northwestern Polytechnical University, Lenovo, HUAWEI	0.9336
2	Tencent-DPPR	Tencent (Data Platform Precision Recommendation)	0.9303
3	Amap-CVLab	Alibaba AMAP, Alibaba DAMO Academy for Discovery, Adventure, Momentum and Outlook	0.9250
4	HH-Lab *	Huazhong University of Science and Technology (Visual and Learning Representation Group)	0.9127
5	maskrcnn_text *	Huazhong University of Science and Technology (Media and Communication Laboratory, Text detection)	0.9102
Baseline		**Meituan Dianping**	**0.9001**

TABLE IV: RESULTS SUMMARY FOR THE TOP-5 SUBMISSIONS OF TASK 4.

Ranking	Team Name	Affiliation	N.E.D
1	Tencent-DPPR	Tencent (Data Platform Precision Recommendation)	0.8150
2	SANHL_v1	South China University of Technology, The University of Adelaide, Northwestern Polytechnical University, Lenovo, HUAWEI	0.8144
3	HH-Lab *	Huazhong University of Science and Technology (Visual and Learning Representation Group)	0.7943
4	baseline_Beihang *	Beihang University	0.7661
5	SECAI *	Institute of Information Engineering, Chinese Academy of Sciences, University of Science & Technology Beijing	0.7437
Baseline		**Meituan Dianping**	**0.7298**

* means student contestant

微信扫码关注技术团队公众号

tech.meituan.com
美团技术博客

新年
快乐