Recommender Systems and Information Fusion in Spatial Crowdsourcing

By

Daniel Alejandro Garcia Ulloa
Doctor of Philosophy

Computer Science and Informatics

Recommender Systems and Information Fusion in Spatial Crowdsourcing

By

Daniel Alejandro Garcia Ulloa
Ph.D., Emory University, 2017

Advisor: Li Xiong, Ph.D.
Advisor: Vaidy Sunderam, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the Graduate School
of Emory University in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2017

Abstract

Recommender Systems and Information Fusion in Spatial Crowdsourcing
By Daniel Alejandro Garcia Ulloa

Spatial Crowdsourcing (SC) refers to a series of data collection mechanisms where a set of users with a sensing or computing device are asked to perform a set of tasks at different locations and times. In this work, we explore some of the challenges that arise with SC and propose some solutions. *These challenges concern a proper recommendation of tasks to users in such a way that they maximize their expected utility while at the same time maximizing the probability that all the tasks are performed.* The utility for the users can be based on the tasks the expected reward they are planning to obtain, and the distance to the assignments. These aspects can be predicted through tensor-factorization techniques. A high-paying assignment might be far from a user, while a low paying assignment is nearby. Depending on the users' preference, we seek to recommend a set of tasks that maximize the user's utility. On the other hand, we also want to maximize the sum of probabilities that the tasks are performed, considering the interdependencies between users. We define the system utility as a convex linear combination of the user and the task utility and suggest approximation methods to recommend the tasks that yield the highest system utility.

We also deal with the problem of truth inference, which focuses on integrating the responses from a mobile crowdsourcing scenario and determining the true value. Many times, the answers from a mobile crowdsourcing scenario are noisy, contradicting, or have missing values. We developed a recursive Bayesian system that updates the reputation model of the users, the probability that the users were in the correct time and location, and the probability that the events are true. We further enhanced this algorithm using a Kalman filter that predicts the true state of the event at each timestamp using a hidden event model and which is updated with the reports from the users. Our method was compared against the naive majority voting method as well as other state-of-the-art truth inference algorithms and our method shows a considerable improvement.

Recommender Systems and Information Fusion in Spatial Crowdsourcing

By

Daniel Alejandro Garcia Ulloa
Ph.D., Emory University, 2017

Advisor: Li Xiong, Ph.D.
Advisor: Vaidy Sunderam, Ph.D.

A dissertation submitted to the Faculty of the Graduate School
of Emory University in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2017

Acknowledgments

This work would not have been possible without the help and support from Professor Xiong. Throughout several years she has provided me with invaluable guidance both in group and in individual meetings, and her ideas and knowledge have given shape to this dissertation. Furthermore, Professor Xiong has always encouraged diverse group integration and individual professional development.

The advice given by Professor Sunderam has also given shape to this dissertation. His insightful ideas during our individual meetings have been very helpful to put into perspective our work. Both Professor Xiong and Professor Sunderam have been excellent mentors both for academic and extracurricular activities, and I could not have wished for better advisers.

I had the honor of working in an internship with Dr. Jiang, whose ideas on recommender systems and tensor factorization were an inspiration to a considerable part of this dissertation. His diligence and hard work are examples of extraordinary mentorship.

I would like to give special thanks to Dr. Ho for her insightful ideas and knowledge on tensor factorizations. The advise she provided has helped me dramatically improve the quality of this dissertation.

Finally, I would like to thank my friends and peers that have helped me so much during these years of postgraduate studies.

A mis padres, que me enseñaron que el patrimonio más valioso es la educación,
y a mi esposa, quien ha sido mi inspiración y me ha mostrado apoyo incondicional.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Spatial Crowdsourcing (SC) refers to a series of data collection mechanisms where a set of users with a sensing or computing device (e.g. a smartphone) are asked to perform a set of tasks at different locations and times. It is an emerging field due to the omnipresence of mobile phones and the wide variety of problems that can be solved [32, 67], and the incentives that users have for performing the tasks.

We identify the following three entities in spatial crowdsourcing:

1. *Users* or participants are entities that use a sensing or mobile device to obtain or measure the required data about a subject of interest.

2. *Requesters* are the entities that request data through tasks and then utilize the information acquired by users.

3. *Task server* or tasking entities are responsible for the distribution of tasks to the users and in some applications are also responsible for fusing the data and deliver a unified answer. There are also other applications and models in which the requesters act as a tasking entity.

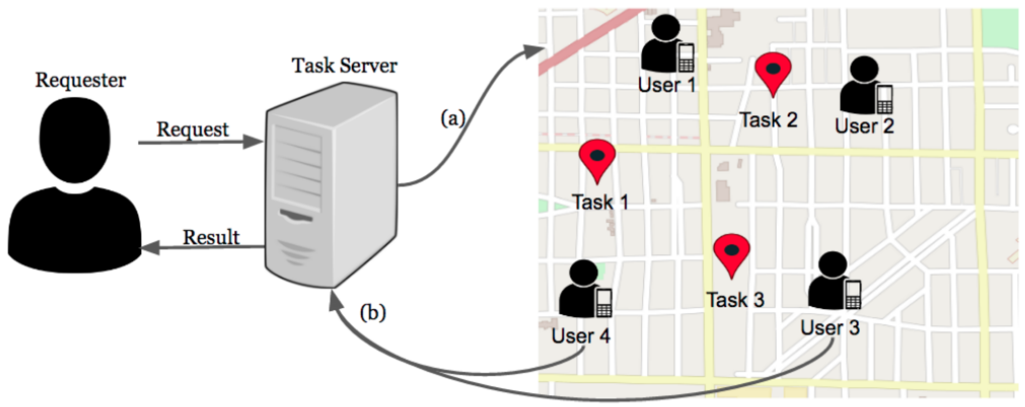


Figure 1.1: General structure of a centralized spatial crowdsourcing scenario. The task server maps users and tasks (a) and integrates the data (b) before sending it back to the requester

Examples of SC include monitoring the price of gas at different gas stations throughout several weeks [77], reporting the traffic at different locations [60, 66], or the availability of parking spots on a street throughout the day [40, 11].

In this work, we focus on **event based, spatial tasks**, mapped with a **centralized** distribution model. *Event based* tasks are triggered when a particular situation occurs. This includes special circumstances such as the presence of a user at a specific location or an ad hoc incident [33]. *Spatial tasks* require the user to be at a specific place in order to fulfill a task. With the increasing use of smart phones with integrated GPS, the number of applications in which tasks are assigned based on the location of participants has also grown [13]. In a *centralized task distribution model*, a task server provides the users with different tasks to perform [41].

Figure 1.1 shows a general structure of SC. A requester sends a SC request to the task server or tasking entity. The tasking entity has then two jobs:

(a) Find an efficient mapping between tasks and users. The mapping is done based on certain criteria which usually relates to the location of the tasks and the users at a specific time. The users are then able to perform the SC tasks and report back to the tasking entity. For some applications, it is desirable to be able to predict the location of the users or the tasks at a future time point before mapping them.

(b) Once the tasking entity receives the answers or reports from the users, in some applications the tasking entity needs to fuse the information and determine which reports are correct, incorrect, or missing, and establish a reliability model for the users. This process is known as truth inference. The tasking entity then sends a unified answer to the requester.

We focus on some of the problems presented in both (a) and (b). For (a), some of the challenges that arise in mapping the tasks and the users include finding an objective function that benefits both the users and the tasks. Several state-of-the-art works focus only on maximizing a utility from the users' point of view so that the users travel the shortest distance or perform tasks that are "on the way" [35, 8, 36] while other focus on fulfilling the maximum number of tasks under budget or time constraints [47, 74]. In most of these cases, the tasks are assigned to users to optimize an objective function, and they do not give any freedom to the user to choose a different task.

For (b), one major challenge that arises in integrating the data from the different users is that the data is incomplete or inconsistent. This is particularly difficult in spatial crowdsourcing since the events can be constantly changing, leading to seemingly inconsistent reports. Another source for inconsistent reports is the users' reliability, since some users can be untrustworthy and report incorrectly. On the other hand, the data could show missing reports because few users where at the specified time and location, or because users might show a lack of participation towards reporting an event. Some state-of-the-art works do not consider events that can be changing over time [16, 64] or model the users' reliability as a single-dimensioned value [46, 53], which does not capture the complexity of their reliability.

In this work, we propose 1) an effective task recommendation system that optimizes both the user and the task utility, and 2) an accurate truth inference system that incorporates user reliability and spatio-temporal characteristics for spatial crowdsourcing.

1.2 Recommender Systems in Spatial Crowdsourcing

As the proliferation of spatial crowdsourcing increases, so does the need to efficiently map the users and the tasks. This is particularly difficult in a spatial scenario since the tasks can be time-sensitive and the location of the users is changing. The major challenges in the mapping depend on the task distribution methodology and the type of tasks.

1.2.1 Current Challenges

Centralized distribution models can also have a **push** or **pull** methodology for mapping the tasks and users. In a *pull* methodology, the task server stores the tasks and allows the users to choose a task according to their skills and their preferences, which can change from time to time. The users have certain independence and can try different tasks on a first-come-first-serve basis [72, 60]. However, research has found that in pull methods, about 33% of the users have difficulties finding an appropriate task [70], and the amount of time spent choosing a task is comparable to that of performing it [84].

On the other hand, *push* methods consist on directly assigning a task to a user, and they have the advantage of eliminating the time spent choosing.

Tasks are usually assigned so as to optimize the mapping between users and tasks according to some criteria, which usually benefits the completion of the tasks [47, 74]. However, this mapping is usually hidden to the user and it gives them no liberty to choose from a variety of tasks that they would be capable of performing.

Recommender systems in spatial crowdsourcing balances the advantages from both methods, by pushing a set of optimized tasks, and allowing the user to pull a task from this reduced set.

In some applications where the tasks are time-sensitive, it is desirable to predict ahead of time some features about the tasks, like their location and the benefit they provide to the users. Predicting the location and benefit of upcoming tasks will allow users to decide ahead of time if they want to perform these tasks, increasing their overall expected utility.

1.2.2 Existing Solutions and Limitations

In terms of mapping tasks and users in spatial crowdsourcing, several works ([18, 56, 8, 47]) assign a path for users under several restrictions (e.g. budget or time restrictions, completing the highest number of tasks). However, they leave no actual options for the users and simply assign tasks to users in such a way that some objective function is maximized. Other works ([28, 35]) consider preferences and skills, but they assume that the users will choose the task that maximizes their utility. In the first case, the utility function that is being maximized is from the tasks' point of view, and in the second case, it is from the users' point of view.

In terms of prediction, the works presented in [3, 57, 38] use Markov models and Bayesian inference to predict the location of the users. In the case of Markov models, raw GPS measurements are used to infer the destination and means of transportation of the users. Historical data is used to model the activities and predict their behaviour. These models could be transferred to moving task prediction, and in the case of events that change their state, Markov models can also be trained with historical data to obtain the probability of a next state given the current one. In our crowdsourcing scenarios we do not assume that the tasks are mobile.

1.2.3 Framework of proposed solution

In a spatial crowdsourcing scenario, the utility for a user is a function on the distance that the user needs to travel to perform a task (the lower the distance, the higher the utility) and the benefit that the user will receive for performing such task (the higher the benefit, the higher the utility). In our crowdsourcing scenarios, we assume that tasks can only be performed once and that any user in the system can perform it. For example, such a crowdsourcing scenario could be taxi or uber[1] drivers (users) looking to pick passengers up (tasks), or freelance workers looking to perform a simple task, as in taskrabbit[2] or gigwalk.[3] On the other hand, the utility for the task is the probability that a task is performed, given their current location, the benefit they provide to the users, and the available users at the time.

In our system, we do not assume that a user will simply choose the task that provides the highest benefit, but instead model the likelihood of choosing a task with a probability distribution that considers the possibility that the user will not perform any task at all. This system more closely reflects real-world situations that are not considered by other models, or that require users to provide private information about themselves [35].

We propose a recommender system that maximizes the expected utility from the users' point of view while at the same time maximizing the sum of

[1]http://uber.com
[2]http://www.taskrabit.com
[3]http://www.gigwalk.com

probabilities that the tasks are performed. These objectives are usually in contention since maximizing the expected utility for the users could mean that low-paying or faraway tasks would not be performed, and maximizing the probability that all tasks are performed might sacrifice the expected utility for the users.

We also propose a method using low-rank tensor approximation as a way to analyze and predict features of the tasks. In particular, we use non-negative CP factorization since it is a helpful method to not just for prediction but also for analysis. Furthermore, tensor factorization techniques have been shown to be very effective for temporal data with varying periodic patterns [24]. In chapter 3 we designed and performed a case study that uses a real-world dataset from a crowdsourcing scenario and uses non-negative CP factorization to analyze and predict the tasks.

1.3 Truth Inference in Spatial Crowdsourcing

In a centralized task distribution system, the task assignment can also be done autonomously. An **autonomous task distribution model** is an allocation method in which the participants have access to a set of tasks and they autonomously choose one or more tasks to perform. The participants do not necessarily need to inform the task distributing entity of their decision. This allows the users to perform the tasks they prefer, but it also leads to missing data if the user decides not to report back. In contrast, **coordinated task assignment** aims at improving the quality of the data by optimizing the set of participants recruited to perform tasks. This optimization is based on varied criteria including coverage, quality, sensing costs, and credibility of the reported data [68]. We focus on autonomous task distribution because of the wide variety of applications and because the users do not need to provide sensitive data to the task server.

1.3.1 Current Challenges

Determining the accuracy of reports in spatial crowdsourcing is particularly difficult if the reports correspond to events that might change over time. Sources often present uncertain or even contradicting reports which can result in considerable loss of accuracy [4]. In general, crowdsourcing users may not visit all target locations, and even if they do, they might not necessarily report the state of an event. Simply ignoring the missing reports has repercussions for the accuracy of any truth-discovery method [22, 30].

Truth discovery or fact-finding [79, 54] seeks to integrate data from different sources and determine the true values of target events. It has gained attention due to the wide variety of possible implementations and the increasing interest in crowdsourcing applications.

1.3.2 Existing Solutions and Limitations

While there have been many recent studies on truth discovery in various crowdsourcing applications (e.g. crowdsourced labeling of online websites [17]), most of them do not consider the sources of data to be mobile, or do not consider that the events could change their state over time [87, 30]. Few methods consider spatial events but do not handle streaming data, and would need to re-run the algorithm each time new data arrives [63, 11]. Other methods (e.g. [55, 86]) handle streaming data and changing truth, as in the case of weather and stock prices, but do not consider the location of the sources and their mobility.

1.3.3 Framework of proposed solution

We consider a set of spatio-temporal events which can be constantly changing from a true state to false state and vice versa, while a set of users report the state of these events at different times. Our goal is to determine or estimate the true state of the event at each time point based on the users' reports.

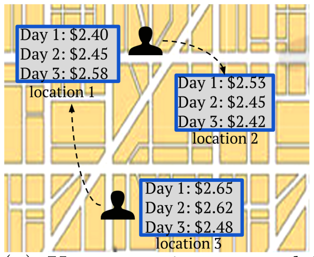
Consider a running example shown in Figure 1.2. Users could be asked to report if the gas price at specific gas stations is below \$2.50 a gallon. The reports of the users will depend on whether they visit the specific location at the specific time of the event as well as their reliability. A passive user could be at the specific location but decides not to send a report, while a malicious user could send a report about a location different from their own. Additionally, seemingly inconsistent reports for a single event could be correct if they were made at different times. The price of gas at a gas station could be below \$2.50 one day and above \$2.50 the next. Figure 1.2(a) shows this example for location 1 between Day 2 and Day 3. Another motivating example is reporting whether there is high traffic at different locations, the way it is done through the app Waze[4]. In this scenario, the traffic in one location could be correlated to traffic in another location. In sections 4.1 and 5.3 we discuss other applications under the scenario described above, and how this analysis can be extended to include more states.

1.4 Contributions

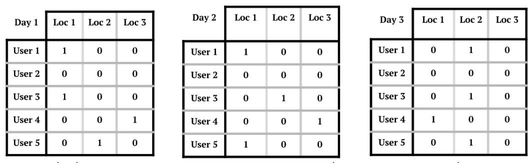
In this section, we briefly discuss the contributions of our work. The contributions in spatial crowdsourcing are both in the mapping between users and tasks and the truth inference problem. We summarize the contributions below.

1. We propose a recommender system in spatial crowdsourcing that aims at maximizing the utility of both the users' and the tasks' point of view. To do this, we:

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



(a) Users moving around locations and sending reports

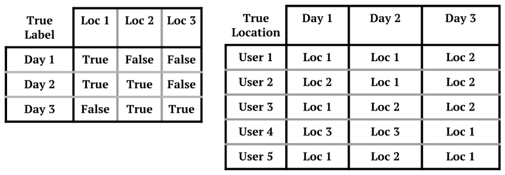


Day 1	Loc 1	Loc 2	Loc 3
User 1	1	0	0
User 2	0	0	0
User 3	1	0	0
User 4	0	0	1
User 5	0	1	0

Day 2	Loc 1	Loc 2	Loc 3
User 1	1	0	0
User 2	0	0	0
User 3	0	1	0
User 4	0	0	1
User 5	1	0	0

Day 3	Loc 1	Loc 2	Loc 3
User 1	0	1	0
User 2	0	0	0
User 3	0	1	0
User 4	1	0	0
User 5	0	1	0

(b) Reports from users (observable).



True Label	Loc 1	Loc 2	Loc 3
Day 1	True	False	False
Day 2	True	True	False
Day 3	False	True	True

True Location	Day 1	Day 2	Day 3
User 1	Loc 1	Loc 1	Loc 2
User 2	Loc 2	Loc 1	Loc 2
User 3	Loc 1	Loc 2	Loc 2
User 4	Loc 3	Loc 3	Loc 1
User 5	Loc 1	Loc 2	Loc 1

(c) True labels of the events and true locations of the users (not observable).

Figure 1.2: Example of spatio-temporal crowdsourced task. Users send true reports ('1') if the price at different locations is less than \$2.50. Based on the reports, our goal is to determine the true labels of the events.

- Formalize the system's utility maximization problem of recommender systems in spatial crowdsourcing by taking into consideration both the users' and the tasks' point of view, and prove it is an NP-hard problem.
- Implement solutions to the problem by proposing algorithms that show the trade-off between utility and speed.
- Evaluate the proposed algorithms both on simulated data and on a real-world case study.
- Propose and implement a method to predict the location and the benefit of tasks in the near-future using tensor factorization techniques and historical data.

2. We propose a solution to the truth inference problem in spatial crowdsourcing to determine the true label of an event from missing or inconsistent data. To do this, we:

    - Present a dynamic graphical model that describes the dependencies of the hidden (true labels of the events, reliability of the users) and observed variables (reports from moving users) in a spatio-temporal setting.
    - Present a recursive Bayesian estimation (BE) method for training the parameters for inferring the true state of the events. Our method incorporates a reliability model for users, which improves as more reports arrive while increasing the accuracy of the model in labeling the state of the event.
    - Further enhance the graphical model with an event model that explicitly describes the spatio-temporal correlations between the events and present a Kalman Filter based approach (BE+KE) for improved inference of the true states.

- Perform experimental validation of the model and algorithms using simulated and real-world data. The experimental results show that our methods are adaptable to the available data, can incorporate previous beliefs, and outperform existing truth discovery methods of spatio-temporal events from crowd sourced data.

Chapter 2

Related Work

In this chapter, we describe previous works in spatial crowdsourcing. The first section is dedicated to SC in general, describing the different types of tasks and distribution models. In the second section, we describe previous works on recommender systems in SC. The last section is a summary of previous works on truth inference in SC.

2.1 Spatial Crowdsourcing

Spatial Crowdsourcing is an emerging topic with a wide variety of possible applications. Tasks can be classified as event-based vs continuous and spatial vs non-spatial, while the distribution models can be classified as centralized vs decentralized vs hybrid, push vs pull, and autonomous vs coordinated. In this section, we emphasize event-based, spatial tasks with a centralized distribution model since this is the type of crowdsourcing scenario of interest in this work.

2.1.1 Classification of Tasks

We classify the types of tasks according to whether there is a special situation that prompts the user to send data back to the tasking entity or not, and whether the task requires the users to be at a particular location and time to perform it or not. The different types of classifications are independent of each other, and any combination can occur.

Event-based vs Continuous

Event based tasks are triggered when a particular situation occurs, and are initiated when the tasking entity requests information about a specific event or circumstance. For example, the requester or tasking entity might need to know if there are potholes or graffiti at a particular location [64, 33]. In this case, these are events that are only reported if the event is true, and the default state is false. Other event-based tasks that are also spatial do not have a default state. For example, the tasking entity could ask participants to act as citizen journalists and submit images or other information from a scene of interest when an event occurs [15]. Other event-based tasks that are not spatial include works on sentiment analysis and classification of documents and images [87]. The submitted data can be of several kinds. In our first examples, the reports were just the label of an event, while on the other examples the users could submit images, video, text, and numerical data.

Event-based methods might pose a privacy threat to the users on certain scenarios. Consider for example task-tracing attacks, linking several tasks that the user performed can reveal sensitive data. Sending an image can reveal time, location, type of device, and events in which the user is interested. This information by itself might not be enough to reveal the identity of the user, but linking multiple tasking actions might allow an adversary to trace the selected tasks by the participant and consequently reveal the user's identity or other sensitive attributes [72].

Continuous tasks receive information from the users periodically or frequently. For example, data could be requested every few minutes to monitor the speed of cars on a specific highway [41] or vital signs of a patient can be frequently requested to track the development of an illness [5]. Continuous tasks can pose a privacy threat to users, since sensitive data like their location can be constantly requested, revealing home or work address and life routines.

Spatial vs Non-Spatial

Spatial tasks require the user to be at a specific location in order to perform the task. Examples of spatial tasks include those in which sensors such as GPS and accelerometers are positioned in vehicles to detect road conditions. Some of these tasks run in the background with little or no involvement from the user, and they could be used to detect traffic speed, bumps, inclination, and elevation of the road [25, 60, 41].

Under certain circumstances, time is also an important factor, since the tasks change over time. Consider for example if the task is to report a traffic accident, or if a gas station remains functional after a natural disaster [77]. Some tasks could ask the users to search for the best prices located at different stores and report them to provide other users with the best prices around [19, 6]. In all of these cases, seemingly inconsistent reports might be correct when the time dimension is considered.

Examples of spatial tasks that are not time-dependent include reporting whether a restaurant is pet-friendly, or taking pictures of a landmark from different angles [53]. In these cases, inconsistent data might be the product of malicious users, mistakes, or misperceptions. It is therefore important to consider the reliability of the users for truth inference.

Spatial tasks can pose a risk in privacy. Location-based attacks can lead to the disclosure of sensitive locations such as home or work addresses and eventually to the user identification [51]. For example, a location could be considered as a home if it is visited frequently by the same user at night [7]. The trajectory data can be also used to infer the individuals' life patterns (i.e. schedules or lifestyles) [82, 31]. One way to ameliorate the privacy risks is to only send data when performing the task (i.e. event-based tasks as opposed to continuous tasks), not send the user's specific location (i.e. spatial-cloaking [48]) or send the data with k-anonymity [37, 43]. In our truth inference scenario, we allow users to send reports from a different location (i.e. noisy data).

Non-spatial tasks do not require the user to be at a specific location to send a report. This includes the tasks related to crowdsourcing platforms such as the Amazon Mechanical Turk[1] and Crowdflower[2] [69, 27]. For example, in [80], users are asked to report whether two descriptions of product match (e.g. iPad 2 vs iPad second generation).

2.1.2 Classification of Distribution Models

Task distribution models refer to the way in which the tasks and the users are mapped. Similarly to the task classification, the dimensions here are also independent and any combination can occur.

Centralized vs Decentralized vs Hybrid

In a **centralized** task distribution system, the tasking entity is a central server. For example, in a party thermometer application, a central server could choose a set of participants attending an event or party, and request that they rate it. These ratings could serve other users who are considering attending this event [15].

The CarTel model [41] is a classic example in which opportunistic sensors were placed in vehicles in the cities of Boston and Seattle for over a year, and they reported to a central server diverse parameters such as location and speed. The central server is provided with a database and an interface

[1]https://www.mturk.com

[2]https://www.crowdflower.com

that allows users to modify certain parameters about the way the data is collected, as well as to visualize the data from the sensors. Further data processing can be achieved once the data has been gathered, enabling for example, traffic monitoring in different locations at different times of the day, as well as performing diagnostics on driving patterns.

A network infrastructure can also be used as a central entity as opposed to having a single server. This could be used for speeding up computations or to avoid having a single point of failure [44].

In a **decentralized model**, each participant can become a tasking entity and decide either to perform a task or pass it forward to other participants who might be better-suited to fulfill the task. This decision would be based on certain attributes of other participants such as location, abilities, or the available hardware in her device. A decentralized recruitment model is proposed in [76] which notifies qualified participants of a forthcoming crowdsourcing activity.

A **hybrid model** includes parts of the centralized and the decentralized models. In this scheme, a central server and a set of participants who act as tasking entities build the task management core. A bubble scheme [59] requires a central server to maintain control of the sensing tasks, which are allocated mostly in a decentralized way. In this model, a task is defined and broadcasted in a particular location of interest by a participant. The task is registered in the server and other participants who move into the location of interest are signaled by the central server and become bubble carriers. These carriers can broadcast the task and can also fulfill them and report the sensed data to the server.

Push vs Pull

Push model based tasks are initiated by a tasking entity via pushing the tasks on the users' devices. The tasks are assigned by the tasking entity in such a way that a certain criteria is optimized and could depend on other factors such as the location of the participants or the time of the day. The optimization function is usually hidden from the users and it mostly corresponds to the efficient fulfilment of the tasks. For example, in the model proposed in [74], tasks are pushed to the user's device given their location so that they can report hyperlocally (e.g. the amount of rain in their area).

One of the disadvantages of push models is that the users are not given the liberty to choose between different tasks. The tasking entity usually does the assignment to maximize the fulfilment of the tasks with little or no consideration to the user's preferences.

Pull models based tasks are queried and downloaded by the users at an arbitrary time or location. A pull based task model can be found in [72], where a set of tasks are stored in a central tasking entity and the users pull this information and decide which tasks to perform. The decision could be based on different criteria such as preferences, location, or the sensors' capabilities. Nericell [60] represents another pull model example, in which the task of opportunistically detecting the road conditions such as potholes, traffic, and noise, depend on the participant's driving route and their smart phone's sensors.

An advantage of pull models is that the users can choose the task they prefer, and maximize their utility based on their own criteria. As a disadvantage, the users need to spend time browsing through several tasks before deciding which one to perform. Furthermore, some tasks are more attractive than others, and will be more likely performed, leaving others tasks unfulfilled.

Autonomous vs Coordinated

Autonomous task selection is an allocation method in which the users have access to a set of tasks and they autonomously choose one or more tasks to perform. The users do not necessarily need to inform the task distributing entity of their decision. The lack of coordination and global optimization for distributing the tasks can decrease the efficiency with respect to sensing cost or global utility.

Another major drawback of autonomous task selection is that it can generate bias in the obtained information. For example, people in urban areas might be more inclined to participate in a sensing task due to the greater presence of mobile devices [1].

Coordinated task assignment aims at improving the quality of the sensed data by optimizing the set of participants recruited to perform tasks. This optimization is based on varied criteria including coverage, quality, sensing costs, and credibility of the sensed data [68].

2.2 Recommender Systems in Spatial Crowdsourcing

Given the rising popularity of crowdsourcing in recent years, there has been an increasing need for having recommender systems in crowdsourcing [34]. Schnitzer et. al. [71] did a study on on the demand for task recommendations in crowdsourcing to avoid assigning the incorrect tasks to users, which would leave both the requester and the users dissatisfied. A recommender system for crowdsourcing based on collaborative filtering is proposed in [58], where the authors use implicit feedback (e.g. tasks shown and tasks performed) to predict the preferences of the users. Other works also predict the user's preferences, either using explicit and implicit feedback from tasks performed [2], or through analysis of their social networks to match their preferences to the characteristics of tasks [20].

2.2.1 Task recommendations

Previous works have addressed the problem of recommending or assigning tasks that are along the path of a moving user. In [8], the authors predict the trajectory of the users, and their objective is to recommend the tasks that are likely to lie along their routine commute. The problem of Maximum Task Assignment is addressed in [47]: given time slices, assign tasks to users so that the sum of tasks assigned throughout all the time slices is maximized. Other works consider tasks that have an expiration time and have as an objective to assign a path for a user that maximizes the number of tasks performed along the way [18, 56]. Several of these problems can be solved through costly but exact algorithms, or through approximating heuristics, that are faster but are not guaranteed to provide the global optimum [29].

Other works in spatial crowdsourcing recommend the closest task without considering the users' preferences. In [10], the authors created a crowdsourcing platform to answer local questions, while in [74], the objective is to maximize the number of tasks performed under budget constraints and dynamic arrivals of users and tasks.

From the users' point of view, the model presented in [35] and [36] considers a context for the users and their objective is to find a set of tasks that maximize the expected commission while maintaining differential privacy for the users. Fonteles et.al.[28] proposed a framework that considers an area of interest for the user, a reward, and similarity between the current tasks and the tasks the user has previously performed.

From the tasks' point of view, the work presented in [45] considers a model that assigns tasks to workers in such a way that it minimizes the number of task assignments while achieving a target reliability. Their work uses low-rank matrix approximations as a powerful predicting tool both to assign tasks to users and to infer the correct answers.

2.2.2 Task prediction

Task prediction in event-based, spatial crowdsourcing depends on the type of event of interest. For example, in the case of events that have a changing state (e.g. the number of available seats at a coffee shop), a Markov model could be used [62]. In other cases, when the event depends on the time and not just on the previous state of the event, time series prediction methods like Holt-Winters or ARIMA have been used [65].

Tensor factorization has been successfully applied in several different works due to its flexibility and ability to incorporate different contexts. In particular, non-negative factorization such as CP or CP-APR [12] can be used to extract meaningful concepts which are concise and easily understood [39]. In this work, we use the CP tensor factorization to predict and analyze features of the tasks. Previous works have used tensor factorization as a recommender system [84], while other works have used this technique to extract meaningful information and predict future behaviours. In [24], the authors use non-negative CP factorization to analyze author-conferences relationships throughout the years and predict what author will publish on a conference in the forthcoming year(s). Other works ([9, 61]) have also used tensor factorization to extract and predict the relationship between different entities in text data with a highly scalable approach.

In recommender systems, Yuen et. al. [84] proposed a framework that uses the search history and the previously performed tasks, together with a 5-point integer scale to recommend tasks via matrix factorization. In general, tensor or matrix factorization has been used as a prediction tool and in particular for recommender systems due to its efficiency, and in the case of non-negative factorizations, its interpretability.

2.3 Truth Inference Algorithms

The process of inferring the correct answer from a set of data obtained through crowdsourcing is known as truth inference. Most algorithms in crowdsourcing include also a method to determine the reliability of the users, which can be helpful for future tasks.

2.3.1 General Classification of Truth Inference Algorithms

Truth discovery methods in spatial crowdsourcing can be classified into iterative, optimization-based, and probabilistic graphical models, although overlaps are possible [54]. In the case of iterative methods, Dong et. al. [22] present a novel method that consider a possible relation between the sources where the value provided by one source is copied by other sources. They use an iterative Bayesian approach to determine these dependencies and infer the true value. Truthfinder [83] presents an approach to use the relationship between different facts and the reliability of the sources of information to determine the true facts from a large amount of conflicting data. An optimization method is provided in [55] where they develop an optimization problem to infer both source reliability and trustworthiness of the information. A probabilistic graphical model can be found in Zhao et.al [85], where the authors proposed an unsupervised Bayesian method that takes advantage of the generation process of false positives and false negatives to infer the true records and the source quality from different databases.

2.3.2 Truth Inference in Spatio-temporal Scenarios

With respect to methods on spatio-temporal crowdsourcing, Wang et.al. [77] proposed an expectation-maximization algorithm that is specifically interested in short-lived crowdsourcing campaigns, where there is no time to build a reputation for the users. Their applications on social sensing deal with events that usually do not vary much over time (e.g. open gas stations after a disaster). Wang et.al [79] have developed an algorithm based on expectation maximization to determine which observations were correct and which ones were not, with no prior knowledge of the source reliability nor about the correctness of prior individual observations. A posterior work [78] includes a sensitivity analysis of the optimal expectation - maximization estimator to understand the accuracy trade-offs in source and claim credibility in social sensing applications. Another approach for dealing with spatial data is the Truth for Spatial Events algorithm (TSE) [64], where not only the truthfulness and reliability of the sources is determined, but also the probability that the users visited locations to improve the accuracy. Since there are some similarities between this work and ours (although our work differs from TSE since we also consider that the state of the events are changing as a function of time), TSE is one of the methods we use to compare with our own.

Most methods of truth inference in spatial crowdsourcing do not consider that the events have a probability model that determines the label of an event. For that reason, we introduce a method that uses the Kalman filter to use the event model and get a higher accuracy by predicting the label of the event using an event model, and combining that data with the information obtained through crowdsourcing.

The Kalman Filter The Kalman filter is a recursive algorithm that uses the available, noisy observations and produces an estimate for the current, hidden state [75, 42]. It can run in real time and consists of two phases: prediction and correction. In the prediction phase, it uses the observations from previous time-steps as input for a system model to predict the state of the system. In the correction phase, the filter trains and uses a parameter (called Kalman gain) to combine the prediction from the previous phase, and the current observations. The Kalman filter has been successfully implemented in applications such as simultaneous localization and mapping, and monitoring web browsing behavior [14, 26].

Chapter 3

Recommender Systems in Spatial Crowdsourcing

In this chapter, we present our recommender system as a task distribution system that combines the benefits of push and pull methods. We formulate the problem and present the details of the model before introducing our proposed solutions. We then present our simulation results and the real-world case study.

3.1 Problem Formulation

The formulation of our problem is done through both the user and the task point of view, and the objective of the recommender system will be to maximize the utility of both.

3.1.1 Users' point of view

Consider the following spatial crowdsourcing scenario, where a set of n_u users or workers are asked to perform n_s tasks at a particular time. Let $U = \{u_1, u_2, \cdots, u_{n_u}\}$ be the set of users and $S = \{s_1, s_2, \cdots, s_{n_s}\}$ be the set of tasks. Each user u_i can only choose and perform one task in the next time-stamp, and each task can only be performed by one user. Each task provides the user with a different utility, which depends on different factors, such as the distance between the user and the task, denoted by dist(u_i, s_i), and a certain payment or "benefit" for performing the task. In many applications, a task will yield the same benefit regardless of the user that performs it, but for the general case, we denote the benefit for user u_i for performing task s_j as $b(u_i, s_j)$. The exact benefit might not be known ahead of time, and users might have different preferences. Some users might prefer doing tasks that are nearby, while others would rather perform a faraway task that yields a higher utility. One of our objectives is to maximize the utility for each user given the distance to the tasks, the expected benefit, and the user's preferences between distance and benefit. If we denote the probability that user u_i performs task s_j as $P(u_i \to s_j)$ and its utility as $Y(u_i, s_j)$, then the expected utility for user u_i for performing task s_j is

$$E(u_i, s_j) = P(u_i \to s_j)Y(u_i, s_j) \tag{3.1}$$

3.1.2 Tasks' point of view

We are also interested in maximizing the probabilities that the tasks are performed. On the one hand, the more users we recommend a task, the higher the probability that some user will choose and perform this task. On the other hand, if we recommend a task to too many users, other tasks will not be recommended, and will have a lower probability of being performed. Furthermore, the users' and tasks' point of views are opposing since tasks can only be performed once, and recommending a task to several users increases its chance of being chosen, but it also decreases the probability for each individual user to perform the task, effectively decreasing the value of $E(u_i, s_j)$. The probability that a task is performed will be simply denoted as $P(s_j)$ and is equal to the sum of probabilities that a user u_i performs it.

In other words,

$$P(s_j) = \sum_{u_i \in U} P(u_i \rightarrow s_j) \tag{3.2}$$

3.1.3 System's point of view

From the system's point of view, our objective is to maximize both the user and the task points of view by recommending the appropriate tasks to the users. For each user u_i we want to obtain a set of tasks $R_i = \{s_{(1)}, s_{(2)}, \cdots, s_{(n_r)}\}$ with $n_r < n_s$ such that the sum of expected utility for the users and the sum of probabilities that the tasks are performed, is maximized. Additionally, the system might have different priorities for each user (e.g. "premium users"). We denote the user's priority as λ_i for $i = 1, ..., n_u$ and $\sum_i \lambda_i = 1$. The system might also have different priorities for each task, and we denote these as γ_j for $j = 1, ..., n_s$ and $\sum_j \gamma_j = 1$. Using the previous notation, our goal is to find $R = \{R_1, \cdots, R_{n_u}\}$ such that

$$f(R) := \alpha \sum_{i=1}^{n_u} \sum_{s_j \in R_i} \lambda_i E(u_i, s_j) + (1-\alpha) \sum_{j \in J(R)} \gamma_j P(s_j) \tag{3.3}$$

is maximized, where $J(R) = \{j | s_j \in \bigcup_{i=1}^{n_u} R_i\}$ and $\alpha \in [0, 1]$ is a parameter that denotes the importance given to the users' point of view and can vary from application to application.

Figure 3.1 shows an example[1] where $n_u = 4$ users are asked to perform $n_s = 3$ tasks at different locations. The distance between the users and the tasks, and the benefit for each user for performing a task is represented in parenthesis. Assuming we give two recommendations to each user ($n_r = 2$), a possible solution would be the sets of recommendations $R_1 = \{s_1, s_2\}$, $R_2 = \{s_1, s_2\}$, $R_3 = \{s_1, s_3\}$, and $R_4 = \{s_1, s_2\}$, represented with solid lines in the figure. Assume that $R = \{R_1, R_2, R_3, R_4\}$ is the optimal solution, and

[1] Not to scale

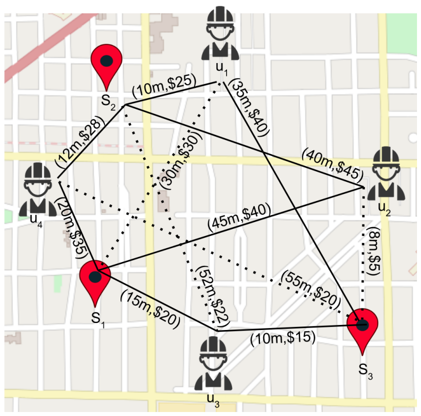


Figure 3.1: Spatial Crowdsourcing scenario with 4 users and 3 tasks. The weights on the edges represent the distance and benefit, respectively.

notice that in this example user u_2 is not recommended task s_3, even though it is closer. This could be because task s_1 and s_2 provide a higher benefit, and the user values the benefit more than the distance. On the other hand, tasks s_1 and s_2 are recommended to user u_1, instead of task s_3, which provides the highest benefit. This could be because user u_1 values distance over benefit.

Theorem 3.1. *Let* $U = \{u_1, u_2, ..., u_{n_u}\}$, $S = \{s_1, ..., s_{n_s}\}$,$\Lambda = \{\lambda_1, ...\lambda_{n_u}\}$, $\Gamma = \{\gamma_1, ..., \gamma_{n_s}\}$, $n_r < n_s$, $\alpha \in [0,1]$, $Y : U \times S \to \mathbb{R}$ *and* $P : \mathcal{P}(U) \times S \to [0,1]$ *with* $\mathcal{P}(U)$ *the power set of* U*. Let* $\mathcal{C}^{n_s}_{n_r}$*, be the set of all combinations of* n_s *choose* n_r*. The problem of finding* $R = \{R_1, \cdots, R_{n_u}\}$ *with* $R_i \in \mathcal{C}^{n_s}_{n_r}$ *such that equation 3.3 is maximized, is NP-hard.*

Proof. To proof NP-hardness, we show a reduction from the generalized assignment problem (GAP), which is known to be NP-hard. The problem is stated as follows: Given a bipartite graph $G = (V, E)$ with partition $V = (A, B)$, $|A| = |B| = 1/2|V|$, and profit and weight function $p, w : E \to [0, \infty)$ and budget $c : A \to [0, \infty)$, find the perfect matching M that maximizes $p(M) = \sum_{e \in M} p(e)$ subject to 1) $w(e) \leq c(a)$ for all $a \in A, e \in M$, and 2) $\forall b \in B \;\; \exists a \in A$ s.t. $e(a, b) \in M$

Given a weighted bipartite graph G, find values for n_u, n_s, and n_r such that $n_r < n_s$ and $n := max(2^{n_u} - 1, |\mathcal{C}^{n_s}_{n_r}|) \geq \frac{1}{2}|V|$, where $\mathcal{C}^{n_s}_{n_r}$ is the set of combinations of $\{1, 2, ..., n_s\}$ choose n_r. Complete graph G to G', with $\frac{1}{2}|V'| = n$ by adding nodes in A, B with weight 0. Solving the problem for G' is equivalent to solving for graph G. Let $U = \{u_1, u_2, ..., u_{n_u}\}$, $S = \{s_1, ..., s_{n_s}\}$ and label the nodes in A' with the elements in $\mathcal{P}(U) \setminus \emptyset$ with $max(w) = w(U)$, and the nodes in B' with the elements in $\mathcal{C}^{n_s}_{n_r}$. Label any remaining nodes with $\emptyset$. Let $\mathcal{A} = \{a \in A' | \cup_{u \in a} u = U, \cap_{u \in a} u = \emptyset\}$ (i.e. a partition of U), and normalize the weights so that $\sum_{a \in \mathcal{A}} w(a, b) = w(U)$. For each $p(a, b)$, find the values $\lambda_{u_i}, \gamma_{s_j}, \alpha, P_{u_i s_j}, Y_{u_i s_j}$ such that $p(a, b) = \alpha \sum\limits_{u_i \in a} \sum\limits_{s_j \in b} \lambda_{u_i} P_{u_i s_j}, Y_{u_i s_j} + (1 - \alpha) \sum\limits_{u_i \in a} \sum\limits_{s_j \in b} \gamma_{s_j} P_{u_i s_j}$. Since the system is over-determined, there is always a solution. Let $R = \{\mathcal{A}\}_1^{2^{n_u - 1} + 1}$and define $f(R) = \sum\limits_{a \in R} \sum\limits_{b \in B} p(a, b)$. Finding the maximum value for $f(R)$ is equivalent to solving the GAP problem. Therefore, this problem in NP-hard.

□

3.2 Model

In this section we present the different elements that comprise our model and in the next section we present our proposed solutions.

3.2.1 Individual Utility Function

Given the distance between the users and the tasks, and the benefit of performing each task, we can build a utility function for each user and task, which represents also their preferences. As mentioned in the previous section, a user might prefer to perform a nearby task, while another user might prefer a faraway task that yields a higher benefit. Our utility function should have a higher value if 1) a user is near a task; 2) the benefit for performing it is higher; and 3) it represent the users' preference. To reflect the notion in 1), consider for example the function

$$D(u_i, s_j) = \frac{\texttt{maxdist} - dist(u_i, s_j)}{\texttt{maxdist}} \tag{3.4}$$

where $dist(u_i, s_j)$ is some distance measure between the user and the task, and `maxdist` is the maximum possible distances between u_i and s_j. Function D is a value between 0 and 1, and the closer u_i and s_j are, the higher the value of D. For 2), since each task provides a different benefit (which could be dependent or independent of the user), we denote the benefit for user u_i for performing task s_j as $B(u_i, s_j)$. We normalize the values of B to be between 0 and 1 to avoid a bias due to larger numbers on the benefits with respect to the distances. We can now model the user's preference as a linear combination of D and B using a scalar ω, which represents the user's preference between the distance and the benefit.

Formally, given functions D and B, $D, B : U \times S \to \mathbb{R}$ and a set of weights $\Omega = \{\omega_1, ..., \omega_{n_u}\}$ with $\omega_i \in [0, 1] \quad \forall i$, we define the individual utility function $Y : U \times S \to \mathbb{R}$ as

$$Y(u_i, s_j) = \omega_i D(u_i, s_j) + (1 - \omega_i) B(u_i, s_j) \tag{3.5}$$

3.2.2 Independent probability of performing a task

Even if a task provides a higher utility than others, we do not assume that the user will choose that task, and might in fact not choose any task at all. For each user u_i, assume we sort the tasks in a non-decreasing way according to the utility they provide to the user, and let $R_i = \{s_{(1)}, s_{(2)}, \cdots, s_{(n_r)}\}$ be the set of ordered n_r recommendations of tasks for user u_i. Let $R_i^+ = R_i \cup \{s_{(nr+1)}\}$ be the set of recommendations *and* the next best task, which was *not* recommended. Define

$$q_{ij} := \frac{Y(u_i, s_j)}{\sum\limits_{s_k \in R_i^+} Y(u_i, s_k)} \tag{3.6}$$

The value q_{ij} represents a "ceteris paribus" probability that user u_i will choose task s_j for all s_j in R_i, and $q_{i(nr+1)}$ is the probability for the "cost of opportunity" for not choosing any task in the set of recommendations R_i. With this definition, given user u_i, $q_{ij} \leq q_{i(j+1)}$ for all $s_j \in R_i$ since they are proportional to their utility, and $\sum_{s_j \in R_i} q_{ij} < 1$, to represent the probability that a user might not choose a task in R_i, while $\sum_{s_j \in R_i^+} q_{ij} = 1$, so q_{ij} is a probability distribution over R_i^+.

3.2.3 Dependent probability of performing a task

The probability that a user performs a task depends also on the decisions that other users take and who gets to choose first. For example, if we only have two users, WLOG we can assume that half the times user u_1 chooses first and half the times user u_2 chooses first. Then the probability that user u_1 performs task s_j would be defined as $P(u_1 \to s_j) = \frac{1}{2}q_{1j} + \frac{1}{2}(1 - q_{2j})q_{1j} = q_{1j}(1 - \frac{1}{2}q_{2j})$. With three users, we can assume that u_1 chooses first $\frac{1}{3}$ of the times, chooses second $\frac{1}{3}$ of the times and chooses third $\frac{1}{3}$ of the times, so $P(u_1 \to s_j) = \frac{2}{6}q_{1j} + \frac{1}{6}(1 - q_{2j})q_{1j} + \frac{1}{6}(1 - q_{3j})q_{1j} + \frac{2}{6}(1 - q_{3j})(1 - q_{2j})q_{1j} = q_{1j}(1 - \frac{1}{2}(q_{2j} + q_{3j}) + \frac{1}{3}q_{2j}q_{3j})$. The probabilities for $P(u_2 \to s_j)$ and $P(u_3 \to s_j)$ are analogous. In general, for n_u users, we write

$$P(u_i \to s_j) = q_{ij} \left(\sum_{z=1}^{n_u} \frac{(-1)^{z-1}}{z} \sum_{c \in \mathcal{C}_{z-1}^{(-i)}} \prod_{k \in c} q_{kj} \right) \tag{3.7}$$

where $\mathcal{C}_{z-1}^{(-i)}$ is the combinations of the set $\{1, 2, ..., n_u\} \setminus \{i\}$ choose $z - 1$. Continuing with the examples, for four users, $P(u_1 \to s_j) = q_{1j}(1 - \frac{1}{2}(q_{2j} +$

$q_{3j}+q_{4j})+\frac{1}{3}(q_{2j}q_{3j}+q_{2j}q_{4j}+q_{3j}q_{4j})-\frac{1}{4}q_{2j}q_{3j}q_{4j})$. This probability reflects the notion that the more users we recommend task s_j, the lower the probability that each individual user will have of performing it.

3.2.4 Probability that a task is performed

As defined in section 3.1, the probability that a task is performed is the sum of probabilities that each user performs it (equation 3.2). Combined with equation 3.7, we obtain the following formulation:

$$P(s_j) = \sum_{z=1}^{n_u} (-1)^{z-1} \sum_{c \in \mathcal{C}_{z-1}} \prod_{k \in c} q_{kj} \tag{3.8}$$

where $\mathcal{C}_{z-1}$ is the combinations of the set $\{1, 2, ..., n_u\}$ (without removing any i) choose $z-1$. Continuing with the example for three users, $P(s_j) = \sum_{i=1}^{3} P(u_i \to s_j) = q_{1j} + q_{2j} + q_{3j} - (q_{1j}q_{2j} + q_{1j}q_{3j} + q_{2j}q_{3j}) + q_{1j}q_{2j}q_{3j}$

3.3 Methods

In this section, we describe our proposed solutions for finding the best recommendations.

3.3.1 Depth First Search Algorithm

This method goes through all the possible combinations of recommendations to users to find the maximum value for $f(R)$. Figure 3.2 shows a tree structure where each level of the tree corresponds to the possible recommendations for a user, based on the recommendations for a previous user. Since we are checking for all combinations, the order of the users is not relevant. The leafs of the tree are all the possible solutions, and this tree structure shows that the solution space is of size $\binom{n_s}{n_r}^{n_u}$. Since we are checking all solutions, the order of this algorithm is $\mathcal{O}(\binom{n_s}{n_r}^{n_u})$ We can actually calculate the current value for $f(R)$ at each of the nodes in the tree. However, in the next level of the tree, the value of $f(R)$ can increase, decrease, or remain the same. We state that in the following proposition:

Proposition 3.2. *Let $R = \{R_1, \cdots, R_n\}$ for some $n < n_u$, and let $f(R)$ be the system utility function defined in equation 3.3. Then, the value of $f(R) - f(R \cup \{R_{n+1}\})$ is either positive, negative, or zero.*

Proof. WLOG, assume the system only has two users $U = \{u_0, u_1\}$ and $R_0 = \{(s_j)\}$ and $R_1 = \{(s_k)\}$ for some $s_j, s_k \in S$. The following analysis can easily be extended to $n_r > 1$. Assume for simplicity that all users and tasks have the same priority (i.e. $\lambda_i = 1, \gamma_j = 1 \forall i, j$). For u_0, $f(R = \{R_0\}) = \alpha q_{0j} Y_{0j} + (1-\alpha) q_{0j}$. Assume $s_j \neq s_k$. Then $f(R = \{R_0, R_1\}) = \alpha q_{0j} Y_{0j} + (1-\alpha) q_{0j} + \alpha q_{1k} Y_{1k} + (1-\alpha) q_{1k} = f(R_0) + f(R_1)$. Since $f(R_1) \geq 0$ then $f(R = \{R_0\}) \leq f(R = \{R_0, R_1\})$.

On the other hand, if $s_j = s_k$, then $f(R = \{R_0, R_1\}) = \alpha(q_{0j}(1 - \frac{1}{2} q_{1j}) Y_{0j} + q_{1j}(1 - \frac{1}{2} q_{0j}) Y_{1j}) + (1-\alpha)(q_{0j} + q_{1j} - q_{0j} q_{1j})$. If $q_{0j} = 1$ and $Y_{1j} = 0$, then $f(R = \{R_0\}) = \alpha Y_{0j} + (1-\alpha)$ and $f(R = \{R_0, R_1\}) = \alpha(1 - \frac{1}{2} q_{1j}) Y_{0j} + (1-\alpha)$. Since $q_{1j} \in [0, 1]$, then $(1 - \frac{1}{2} q_{1j}) \leq 1$ and $f(R = \{R_0, R_1\}) \leq f(R = \{R_0\})$.

Therefore, adding more users to the system can increase, decrease, or keep the same value for the system utility.

□

Proposition 3.2 makes it harder to devise a method such as branch and bound that could trim some of the branches in the tree and reduce the search space in a significative way.

3.3.2 Greedy Algorithm

Although the exact solution is hard to find efficiently, we consider a greedy algorithm that quickly approximates the solution. The idea of the algorithm

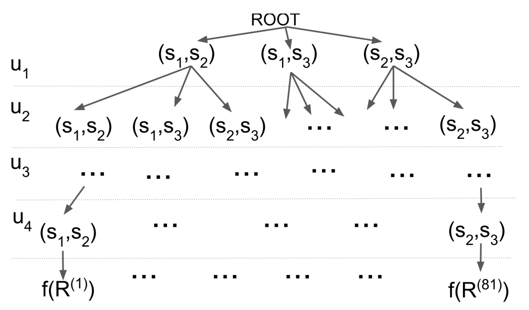


Figure 3.2: Depth-First search of the solution. Here, $f(R^{(1)})$ is the system utility if we recommend (s_1, s_2) to all four users. Since we have 3 tasks, 2 recommendations per user and 4 users, we have a total of 81 possible combinations.

is to recommend to each user the tasks that yield the highest utility without considering dependencies among users. The highest values for the user's utility should also provide a relatively high value for the probability of the tasks being performed. However, it does not guarantee that the maximum value for the system will be found, since it is not considering the dependencies among users. We consider this our baseline approach, and describe the procedure in algorithm 1.
It is easy to see that the order of algorithm 1 is $\mathcal{O}(n_u n_s \log n_s)$

3.3.3 Progressive Algorithm

The greedy algorithm does not consider the interaction between all users, and only relies on the fact that a high utility for the user means a high probability of performing the task and a high overall probability that the task will be performed. In this next approach, we consider the users sequentially, and recommend the best available tasks given the previous users. This approach only considers the dependencies with previous users, and once we have optimized the tasks for a particular user, we do not "look back" and change

Algorithm 1 Greedy Approach

```
1: $R \leftarrow \emptyset$
2: for $u_i$ in $U$ do
3:     for $s_j$ in $S$ do
4:         Obtain $Y(u_i, s_j)$ using equation 3.5
5:     end for
6:     Sort $Y(u_i, s_j)$ so that
       $Y(u_i, s_{(1)}) \leq Y(u_i, s_{(2)}) \leq ... \leq Y(u_i, s_{(n_s)})$
7:     $R_i \leftarrow \{s_{(1)}, s_{(2)}, ..., s_{(n_r)}\}$
8:     $R \leftarrow R \cup \{R_i\}$
9: end for
10: return $R$
```

previous users. The procedure is described in algorithm 2, which is of order $\mathcal{O}(n_u n_s \binom{n_s}{n_r})$.

Recalculate Progressive

One modification to the progressive approach consists in recalculating the values of $P(u_i \rightarrow s_j)$ for all the previous users (without modifying their recommendations), and then calculate $P(u_i \rightarrow s_j)$ for the current user with these updated values. We can now obtain $E(u_i, s_j)$ and $P(s_j)$ for all the tasks and recommend the n_r tasks that yield the highest values for $f(R)$. The only modification in algorithm 2 would be adding a step between step 7 and 8, and recalculate $P(u_\mathbf{i} \rightarrow s_j)$ for all $\mathbf{i} < i$ and for all $s_j \in R$. The order is therefore $\mathcal{O}(n_u^2 n_s n_r \binom{n_s}{n_r})$.

Ordered-Progressive Algorithm

An improvement to the progressive algorithm consists in ordering the users by finding the values of $Y_{ij} \quad \forall u_i, s_j$ and sorting the users based on the indi-

Algorithm 2 Progressive Algorithm

```
1:  $R \leftarrow \emptyset$
2:  for $u_i$ in $U$ do
3:      for $s_j$ in $S$ do
4:          Obtain the value of $Y_{ij}$ using equation 3.5
5:      end for
6:      maxval $\leftarrow 0$, $r^* \leftarrow \emptyset$
7:      for $r$ in $\mathcal{C}_{n_s}^{n_r}$ do
8:          $\hat{R} \leftarrow R \cup \{r\}$
9:          if $f(\hat{R}) \geq$ maxval then
10:             maxval $\leftarrow f(\hat{R})$
11:             $r^* \leftarrow r$
12:         end if
13:     end for
14:     $R \leftarrow R \cup \{r^*\}$
15: end for
16: return $R$
```

vidual utility. The idea behind this approach is that if we optimize first the users with the highest utility for a task, then if that task is in a candidate recommendation set for another user, it would not provide the highest utility, and the algorithm would look for another solution.

To see how the order in which users are optimized affects the utility, and why ordering the users according to their utility is a reasonable heuristic, consider the following example with two users: u_1 and u_2. For ease of notation, let $Y_{ij} := Y(u_i, s_j)$, and let $R^{(i_1,i_2)}$ be the set of recommendations obtained by first optimizing u_{i_1} and then u_{i_2}. Assume, WLOG, that $Y_{1j} \geq Y_{2j}$ for some $s_j \in R_1 \cap R_2$ and that both users have the same priority. If we first run the progressive algorithm with u_1 and then u_2, then

$$\begin{aligned} f(R^{(1,2)}) =& \alpha P(u_1 \to s_j) Y_{ij} + (1-\alpha) P(u_i \to s_j) + \\ & \alpha P(u_2 \to s_j) Y_{2j} + (1-\alpha) \sum_{i=1}^{2} P(u_i \to s_j) \end{aligned}$$

Because of the order of evaluation, $P(u_1 \to s_j) = q_{1j}$, $P(u_2 \to s_j) = q_{2j}(1 - \frac{1}{2}q_{1j})$ (equation 3.7), and $\sum_{i=1}^{2} P(u_i \to s_j) = q_{2j}(1 - q_{1j})$ (equation 3.8). The previous equation can be then written as:

$$\begin{aligned} f(R^{(1,2)}) =& \alpha q_{1j} Y_{1j} + (1-\alpha) q_{1j} + \\ & \alpha q_{2j}(1 - \frac{1}{2} q_{1j}) Y_{2j} + (1-\alpha)(q_{2j} - q_{1j} q_{2j}) \end{aligned}$$

Similarly, if we evaluate first user u_2, then:

$$\begin{aligned} f(R^{(2,1)}) =& \alpha q_{2j} Y_{2j} + (1-\alpha) q_{2j} + \\ & \alpha q_{1j}(1 - \frac{1}{2} q_{2j}) Y_{1j} + (1-\alpha)(q_{1j} - q_{2j} q_{1j}) \end{aligned}$$

Now, consider

$$\begin{aligned} f(R^{(1,2)}) - f(R^{(2,1)}) &= \frac{\alpha}{2} q_{1j} Y_{1j} q_{2j} - \frac{\alpha}{2} q_{2j} Y_{2j} q_{1j} \\ &= \frac{\alpha}{2} q_{1j} q_{2j} (Y_{1j} - Y_{2j}) \geq 0, \end{aligned}$$

since we are assuming $Y_{1j} \geq Y_{2j}$. With this result, we conclude that the order in which the users are evaluated in the progressive algorithm can yield different results, and that ordering the users according to their individual utility represents an advantage over the simple progressive approach.
On the other hand, to order the users according to their individual utility, we need to consider the n_r tasks that yield the highest individual utility. A possible ordering for the users could be done by adding the n_r highest values of Y_{ij} for each user u_i and comparing that value to other users'. In other words, let

$$v_i := \sum_{i=1}^{nr} \texttt{nr} \max_{s_j \in S} Y_{ij} \tag{3.9}$$

where $\texttt{nr}\max$ is the n_r maximum values, and sort U so that $v_{(1)} \geq ... \geq v_{(n_u)}$. Algorithm 3 summarizes the procedure, which is of order $\mathcal{O}(n_u n_s(n_s \log n_s + \binom{n_s}{n_r}))$.

3.3.4 Hybrid Approaches

In these approaches, we group the users according to some heuristic and then optimize.

User Batches

Following a similar line of reasoning as with the progressive algorithm, we can group users in small batches and optimize concurrently. If the batches are small enough, we can use an exact algorithm, or an expensive but accurate algorithm. If we use the exact algorithm and the size of the batch is n_u, then the algorithm is DFS. If the size of the batch is 1, then this approach is progressive. Once we have determined the size of the batch, choosing the users for each batch can be done in different ways such as randomly, or ordered according to their utility, as in ordered-progressive.

Algorithm 3 Ordered-Progressive Algorithm

```
1: $R \leftarrow \emptyset$
2: for $u_i$ in $U$ do
3:     for $s_j$ in $S$ do
4:         Obtain the value of $Y_{ij}$ using equation 3.5
5:         Obtain $v_i$ using equation 3.9
6:     end for
7: end for
8: $V \leftarrow sort(U)$ according to $v_i$
9: for $u_i$ in $V$ do
10:    for $s_j$ in $S$ do
11:        maxval $\leftarrow 0$, $r^* \leftarrow \emptyset$
12:        for $r$ in $\mathcal{C}_{n_s}^{n_r}$ do
13:            $\hat{R} \leftarrow R \cup \{r\}$
14:            if $f(\hat{R}) \geq$ maxval then
15:                maxval $\leftarrow f(\hat{R})$
16:                $r^* \leftarrow r$
17:            end if
18:        end for
19:        $R \leftarrow R \cup \{r^*\}$
20:    end for
21: end for
22: return $R$
```

Instead of choosing the size of the batch first, we can choose how many batches to have, and then cluster the users. We present the following approach:

Clusters

In this approach, we perform any clustering technique, such as k-means, DBSCAN, or hierarchical clustering, based on the position of the users. The clustering technique to use depends on previous knowledge about the distribution of the positions of the users. In real-world scenarios, the users are generally not uniformly distributed, since there are areas of high density and other areas with low population density, so k-means could be a reasonable approach. Once we have clustered the users, we calculate the distance between each task and the center of the clusters. If the distance is less than a threshold τ, then we "assign" the task to the cluster. This way, a task could actually belong to several clusters, or to no clusters if the task is too far. Now we independently optimize each cluster, either using an exact algorithm, or with a faster approximation.
On the other hand, we can also optimize each cluster progressively. In other words, we consider the previous clusters when looking for a solution for the current cluster. The time complexity remains the same, and the only tasks affected are the ones that belong to several clusters. Having dependent clusters does not guarantee finding a better solution than independent clusters, since we could be incorrectly recommending a task to user based on previous clusters, but recommending another task might have yielded a better result. Algorithm 4 summarizes this approach. The order of the hybrid algorithms depend on the clustering or batching technique that was used, as well as on the algorithm used to obtain the recommendations.

Algorithm 4 Hybrid Approach

```
1: $R \leftarrow \emptyset$
2: Partition $U$ into set $\mathbb{P} = \{U_1, ...U_K\}$ using a grouping heuristic such as
   batch or clustering
3: for $U_l$ in $\mathbb{P}$ do
4:     for $u_i$ in $U_l$ do
5:         Find $R_i$ using an exact or a progressive algorithm (2 or 3)
6:         $R \leftarrow R \cup \{R_i\}$
7:     end for
8: end for
9: return $R$
```

3.4 Experiments

In this section, we describe the evaluation metrics that were used to evaluate the different methods, as well as the simulation experiments and the real-world implementation.

3.4.1 Evaluation Metrics

To evaluate the performance of our algorithms, we compare the utility of the different methods. We divide the utility into three separate measures:

1. **User utility**, defined as
   $f_U(R) := \sum_{i=1}^{n_u} \sum_{s_j \in R_i} \lambda_i P(u_i \rightarrow s_j) Y(u_i, s_j)$
2. **Task utility**, defined as
   $f_S(R) := \sum_{\{j | s_j \in \bigcup_{i=1}^{n_u} R_i\}} \gamma_j P(s_j)$, and
3. **System utility**, which is the linear combination of $f_U(R)$ and $f_S(R)$ defined in equation 3.3 with $\alpha = \frac{1}{2}$.

All the utilities are values between 0 and 1, where the closer to 1, the better. However, even the exact algorithm cannot reach such values, since the utility depends on the distance between the users and the tasks, and the benefits each task provides. Even if all the tasks provided the same benefit and the users did not need to travel to perform the task (or equivalently $\omega_i = 0$ for all users in equation 3.5), the utility would still not reach 1 if there is competition among the users, which reduces the probability that a user performs a task (equation 3.7) and therefore also the probability that a task is performed is affected (equation 3.2). The task utility cannot reach 1 either, because even if there are few tasks and many users, there is always a chance that the users decide not to perform the task (section 3.2.2). In fact, we can determine tighter bounds for the utility depending on the data, and have deferred such analysis to the appendix.

Another important performance measure is the computational speed. The reason to use approximation methods as opposed to the exact algorithm is that the exact algorithm does not scale very well, and neither do some approximation algorithms. Therefore, we observe a trade-off between utility and speed. All of our experiments are run in an 8 CPU desktop running Ubuntu 16.04 LTS with Intel(R) Core(TM) i7-6700K CPU at 4GHz and 64 GB in memory.

3.4.2 Simulation experiments

Comparisons against the exact algorithm

On a small scale, we compare the results obtained with the exact algorithm (DFS) to the baseline method (greedy) and our ordered-progressive method. To avoid a bias that would unfairly benefit our method, we assume that all users and all tasks have the same priority. In other words, we set $\lambda_i = 1/n_u$

and $\gamma_j = 1/n_s$ for all i and for all j. On this scale, hybrid approaches perform as well as the ordered-progressive algorithm, so we obviate them from this analysis. We performed 10 simulations and obtained the average utilities (user, task, and system) and the runtime as we vary the number of users, tasks, and recommendations per user.

Comparisons among approximation methods

To evaluate the performance of the approximation algorithms at a larger scale, and to observe the performance of the hybrid approaches, we also ran simulations at a larger scale and observe the performance as we vary the number of users, tasks, and recommendations. Similarly to the comparisons against the exact algorithm, we performed 10 simulations and obtained the average utility and runtime.

3.4.3 Case study

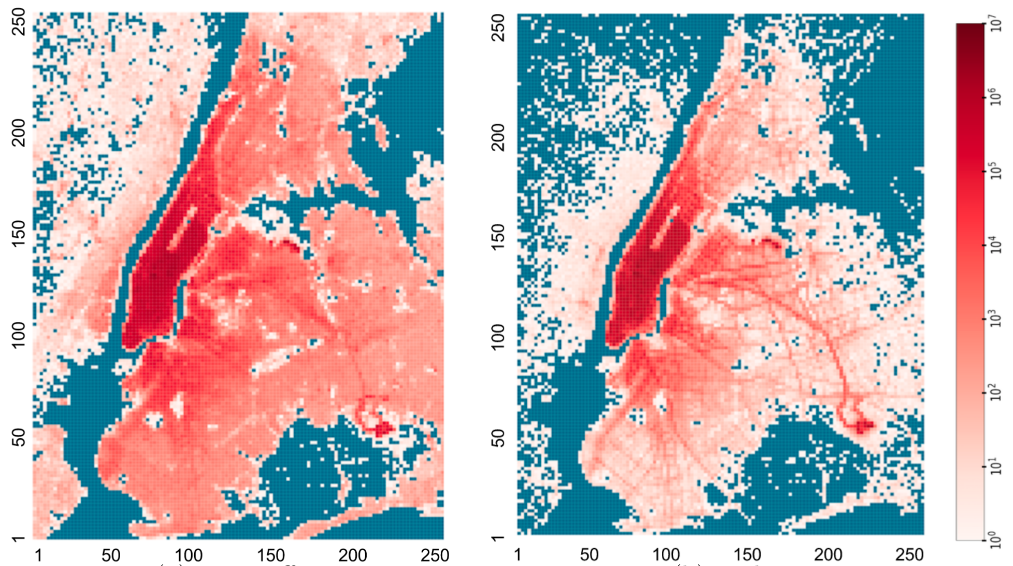
As an application for our spatial task recommender system, consider the case of taxi drivers in a city. In this case, the users are the taxi drivers, and and the tasks consist in picking up people at different locations. The objective of the recommender system is to suggest a small set of pick-ups based on a predicted distance to the task and benefit for providing the service.

Data Set

We conduct our experimental results on yellow taxi trip data of New York City, which is a publicly available dataset[2] and includes pick-up location, pick-up datetime, drop-off location, drop-off datetime, trip distance, fare amount and tip amount. We extracted the data from 01/01/2015 to 06/30/2016 and removed those entries with excessive fare or tip amount which were most

[2]http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

likely caused by an error in measurement (above 4 standard deviations) as well as those trips that included far-away drop-offs. After cleaning the data, we had 209,506,086 trip records between the latitudes and longitudes approximately between (40.5,-74.3) and (41.1,-73.5). We divided this area into a grid of 250 by 250 squares. Each grid is approximately $250m^2$. Figure 3.3 shows a heatmap of the drop-offs (3.3(a)) and pick-ups (3.3(b)) in New York City during this time period. Using the datetimes between drop-offs and pick-ups, and the fare and tip, we calculate the benefit of performing each task as (fare+tip)/(dropoff datetime - pickup datetime), which is an estimate of dollars per minute for performing the task.



(a) Drop-offs (b) Pick-ups

Figure 3.3: Heatmaps of the distribution of taxi drop-offs and pick-ups in New York City between 01/2015 and 06/2016

Approximating the data using low-rank tensor approximations

In many real-world applications, the location and benefit of performing a task is not known ahead of time. For this reason, we propose tensor factorization as a technique to predict both the location of the next tasks and the benefit of performing them. Before describing how tensors can be used to predict these features from the tasks, we start with a few definitions:

Definition 3.3. *The **outer product** of two vectors $u^{(1)}$ and $u^{(2)}$, denoted as $u^{(1)} \otimes u^{(2)}$, is a matrix U such that the coordinates satisfy $U_{ij} = u_i^{(1)} u_j^{(2)}$.*

Definition 3.4. *The outer product of M vectors $u^{(1)} \otimes u^{(2)} \otimes \cdots \otimes u^{(M)}$ produces an **rank-one, Mth order tensor** $\mathcal{X}$ where each element $\mathcal{X}_{i_1,i_2,\ldots,i_M} = u_{i_1}^{(1)} u_{i_2}^{(2)} \cdots u_{i_M}^{(M)}$. The vectors $u^{(j)}$ are called the **modes** of $\mathcal{X}$*

Since matrices are a particular case of tensors, tensor decomposition is thus a generalization of matrix factorization [50, 73]. The CANDECOMP/PARAFAC (CP) decomposition can be considered a generalization of the singular value decomposition (with certain caveats) [50] and it approximates the tensor $\mathcal{X}$ as the sum of R rank-one tensors or **components**:

$$\mathcal{X} \approx \sum_{c=1}^{R} \lambda_c u_c^{(1)} \otimes u_c^{(2)} \otimes \cdots \otimes u_c^{(N)} \tag{3.10}$$

The non-negative CP decomposition imposes the restriction that the entries in the components should be non-negative, which increases the interpretability of the decomposition. Additionally, the decomposition reduces the dimensionality of the data, can help filter out the noise, and can be used to predict the missing data [52, 81].

To appropriately recommend tasks to users, it is desirable to accurately predict ahead of time the demand of tasks and the benefit of performing such tasks. Notice that the predictions are made for all locations at a single datetime point, and in this particular application, the users require some time to perform the task and will very likely end in a different location. At that point, the users would need to ask for a new recommendation. We divided every 24 hour day into 10 minute timeslots (144 timeslots per day) and extracted the day of the week from the pickup datetimes. To make the predictions, we used the historical data between 01/01/2015 and 03/30/2016 to train the model and predict six `objective datetimes` which were all after 03/30/2016.

We built two tensors using the 4 modes (grid_x, grid_y, DoW, timeslot). The first tensor, called the demand tensor (dT) has at each entry the average number of tasks requested, while the second tensor, called the benefit tensor (bT), has the average benefit of performing a task. To avoid a bias due to the differences in measurements between demand and benefit, we standardized the values to a normal 0,1 distribution. We use non-negative CP decomposition as a factorization technique due to its interpretability and conciseness [24, 39]. Figure 3.4 shows the structure of the tensors with four

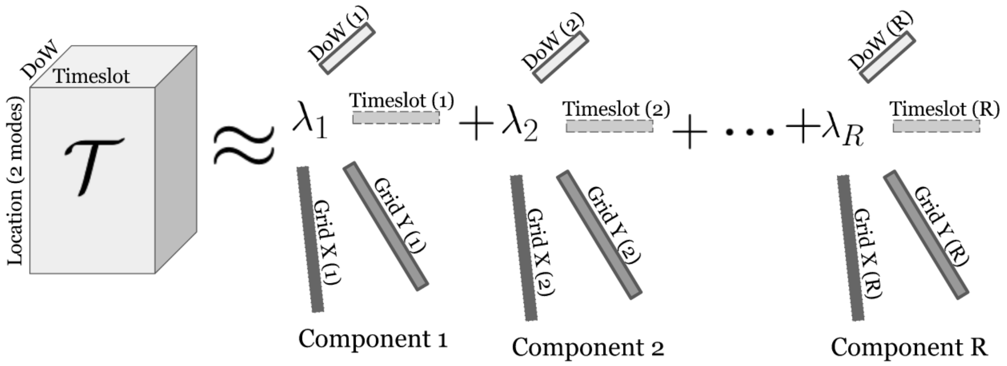


Figure 3.4: CP decomposition of the demand and benefit tensor with R components

modes and its approximation through the non-negative CP decomposition. The tensor is decomposed into the sum of R components. Each component is a rank-1 tensor comprised of the outer product of four vectors. Each vector represent one of the modes. In our case, the vectors corresponding to Grid X and Grid Y are of size 250, the vector for timeslot is of size 144, and the vector for DoW is of size 7. These vectors can also be used to analyze the structure of the data, and can reveal interesting information that is easy to interpret.

To determine the number of components to factorize the tensors, we graphed the approximation error, defined as $||T - \hat{T}_c||_2/||T||_2$, where T is the original tensor and $\hat{T}_c$ is the reconstructed tensor using c components. We chose the minimum number of components after which there was not a significant decrease in the approximation error. Figure 3.5 shows the approximation error as a function of the number of components. We chose 60 components for the benefit tensor (approximation error = 0.3817) and 100 components for the demand tensor (approximation error = 0.3581).

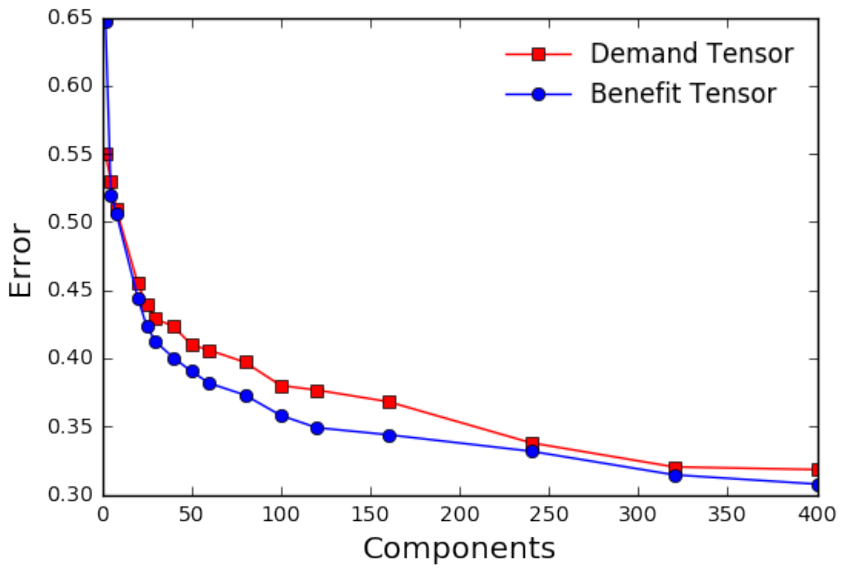


Figure 3.5: Reconstruction error $||T - \hat{T}_c||_2/||T||_2$ as a function of the number of components c

Data analysis using non-negative tensor factorization

In this section, we analyze the data obtained from the vectors after the non-negative parafac decomposition. The basic idea is to plot the first few components of the decomposition (figure 3.6) to obtain insights about the original data.

Figure 3.6(a) shows the first three components of the DoW mode in the demand tensor. The non-negative CP factorization has classified the demand into different types. The first component shows the demand on the weekdays, and corresponds to the people who regularly need the taxi service to commute probably to their job and back home. The second component correspond to the people that need the service mostly on Friday, Saturday and Sunday mostly at night. The third component also corresponds to a higher activity on the weekends, but in this case is for cultural or social activities during the day. This classification is further corroborated by the timeslot mode (figure 3.6(b)), where the first component starts increasing from 6 a.m. to 10 a.m., then it remains somewhat constant and increases again after 5 p.m., obtaining a maximum between 7 p.m. and 10 p.m., and finally decreasing again at night. Component 2 and 3 reflect the people who use the service on the weekends, and in the case of component 2, especially at night. Component 2 starts increasing steadily after 6 p.m. and reaches its maximum value between 12 a.m. and 2 a.m. The third component shows a higher activity between 10 a.m. and 7 p.m.

Compare the first component in this graph to the demand for tasks presented in figure 3.7. The left axis of this figure is the amount of tasks and users on December 1, 2015 (a Tuesday). We observe a similar trend as the interpretation for the first component in the timeslot mode, with steep increases during rush hours.

Figure 3.6(c) and 3.6(d) show the graphs for the first three components in the Grid X and Grid Y modes. We observe the highest values for the three components approximately in grids 92 to 105 for Grid X and 90 to 120 for Grid Y. Compare this to the heatmap for pick-ups presented in figure 3.3(b), which shows the highest activity in Manhattan, and it corresponds to those grid numbers. We can use this information together with the data from the other modes to determine the location, day of the week, and time of the highest demand, and incorporate it to the clustering methods on the hybrid approaches.

An analogous analysis could be done for the benefit tensor, and determine what tasks provide a higher utility. In our prediction model, we focus on those tasks to jointly optimize the prediction and the recommendation and therefore optimize the overall system utility.

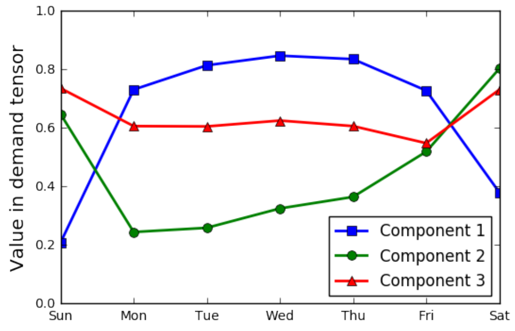
Figure 3.7 shows the benefit (y-axis on the right) throughout the day using the real data from the dataset. We observe the opposite trend as the demand. The higher the demand, the lower the average benefit for performing a task.

Predicting the location and benefit of tasks

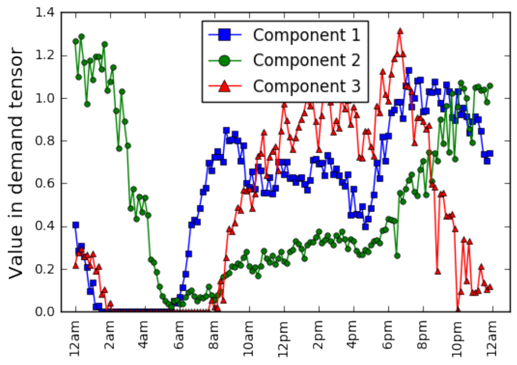
After factorizing $b\hat{T}$ and $d\hat{T}$, for each grid_x and grid_y, we predicted the demand and benefit for the DoW and timeslot corresponding to the `objective datetimes`. In the case of the demand, we rounded the predicted result to the nearest integer.

From the original dataset, we extracted the drop-off locations for `current datetime`, and assumed that these were the position of the users looking for recommendations to perform a task within the next 10 minutes.

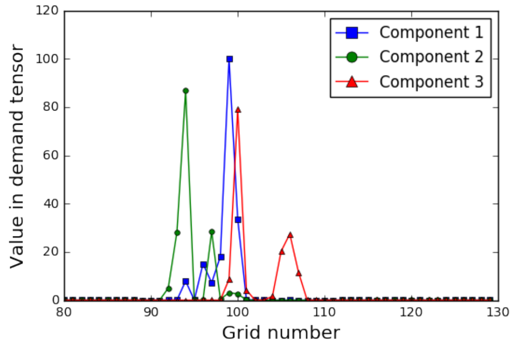
We ran the greedy algorithm on both the predicted data and the real data from the original dataset from the `objective datetime` and obtained recommendation for all users. To run the ordered-progressive algorithm, we used a hybrid approach and considered a moving window of 5 by 5 squares. It is reasonable to assume that users will not be able to travel too far away within the next 10 minutes.



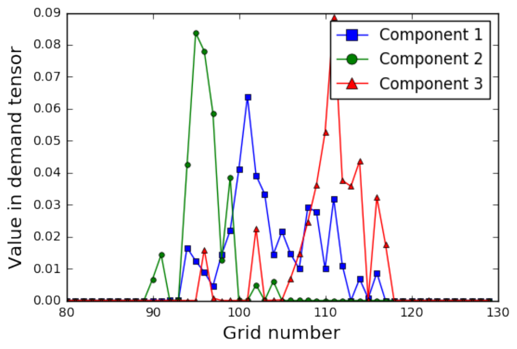
(a) Day of Week Mode



(b) Timeslot Mode



(c) Grid x Mode



(d) Grid y Mode

Figure 3.6: Visualization of the first components of each mode in the demand tensor

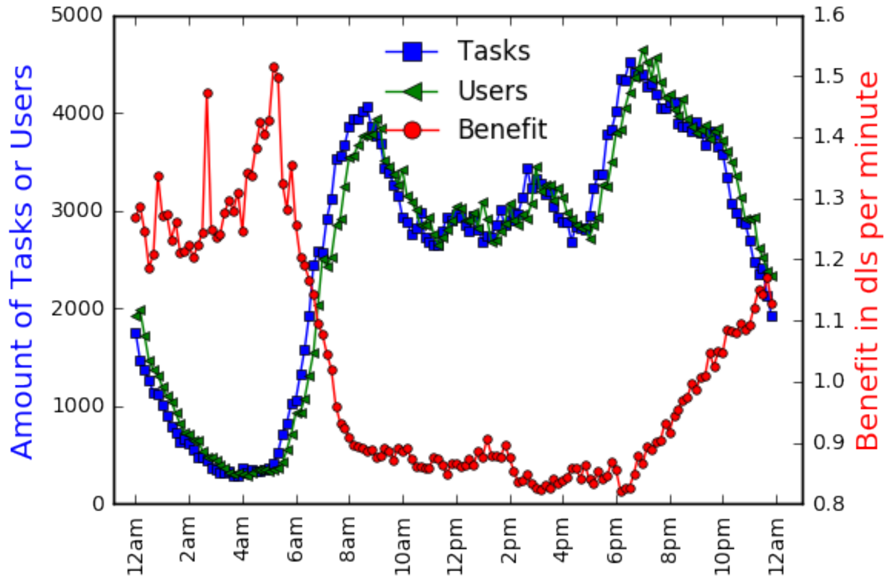


Figure 3.7: Demand and benefit of tasks throughout a day

Likewise, we also ran the algorithms both on the real data and the data predicted by the tensors on other days as well. The `objective datetimes` that were considered are as follows:

1. April 1, 2016 (Friday) at 7:00 p.m.
2. April 28, 2016 (Thursday) at 4:10 a.m.
3. May 11, 2016 (Wednesday) at 11:20 a.m.
4. May 28, 2016 (Saturday) at 2:10 p.m.
5. June 7, 2016 (Tuesday) at 8:30 p.m.
6. June 26, 2016 (Sunday) at 9:00 a.m.

The dates and times were chosen from the test set (between 04/01/2016 and 06/30/2016) in such a way that there would be a variety in the day of the week, time of the day, and from relatively distant days.

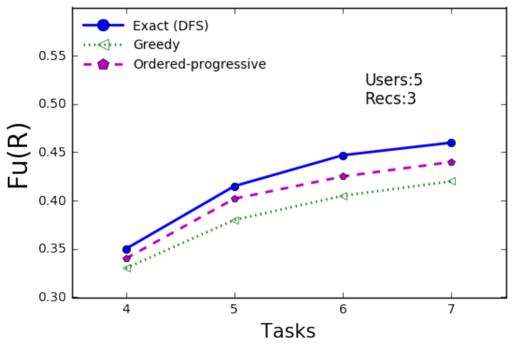
3.5 Results

3.5.1 Simulation Results

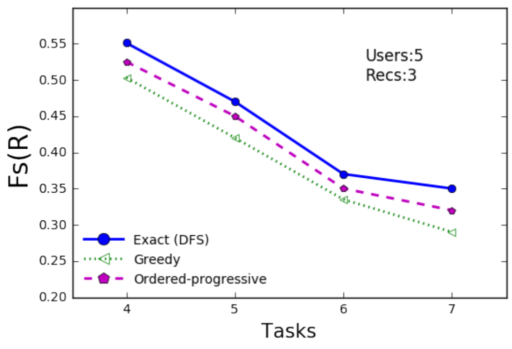
In figures 3.8, 3.9 and 3.10, we observe the change in utility as we vary the users, tasks, and number of recommendations. We perform 10 different simulations and present the median result. In figure 3.9(a) we observe that the user utility decreases as we increase the number of users with a fixed number of tasks. Figure 3.9, show the variation in user, task, and system utility, respectively, as the number of users is increased. In general, the user utility decreases, and the task and system utility increases. We observe the opposite behaviour in figure 3.8 as we increase the number of tasks. In the case of varying the number of recommendations (figure 3.10), there is a slight increase in the utility when using 4 recommendations. In all cases, the ordered-progressive algorithm showed a better performance as an approximation algorithm than the greedy approach. Figure 3.11 shows the log-scale runtime of the algorithms in seconds. The exponential nature of the exact method shows the need to use approximation algorithms for practical solutions.

At a first glance, it was surprising that recalculate progressive performed worse than other algorithms. After a careful look, recalculating previous values for $P(u_i \rightarrow s_j)$ without modifying the recommendations gives an unfair advantage to that tasks that have already been recommended, making the algorithm recommend those tasks again.

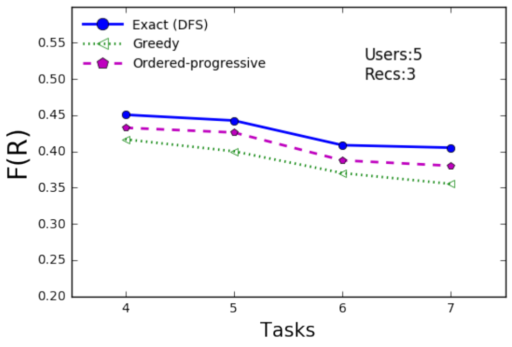
Ordered progressives runs slightly slower than progressive because it needs to order the users according to the tasks (step 5 and 8 in algorithm 3). With respect to the hybrid approaches, cluster runs slightly slower than batch because it needs to run a clustering algorithm, as opposed to batch that simply takes a few users at a time in any order. It is faster to run these algorithms and then run order progressive inside the batches or clusters,



(a) User Utility



(b) Task Utility



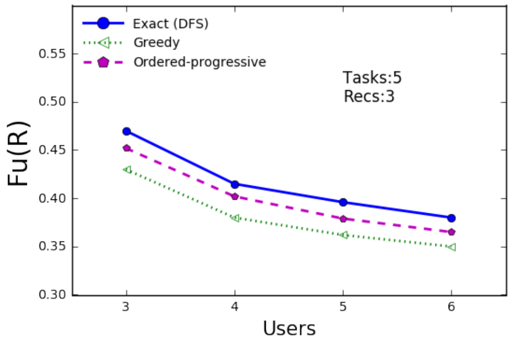
(c) System Utility

Figure 3.8: Utility measures as a function of the tasks.

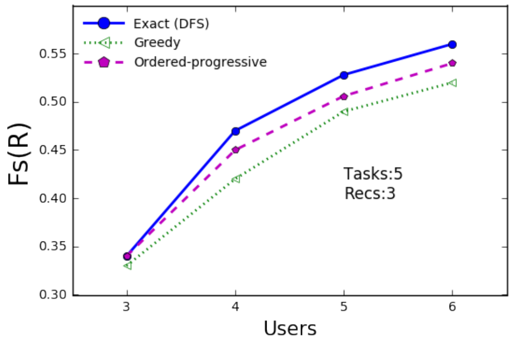
which increases the speed of the ordered-progressive algorithm for real-world applications.

3.5.2 Case Study Results

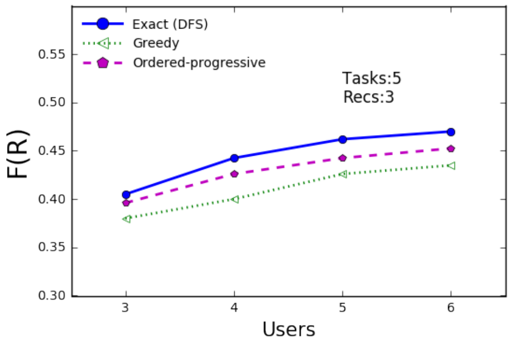
For the case study, figure 3.16 shows the box plots for the greedy and hybrid/ordered progressive algorithm for the user and task utility. Figure 3.16(a) shows the distribution of $E(u_i, s_j)$ for each user (i.e. the user utility), and figure 3.16(b) shows the distribution of $P(s_j)$ for each task (i.e. the task utility). The tensor factorization had a low error predicting the demand



(a) User Utility

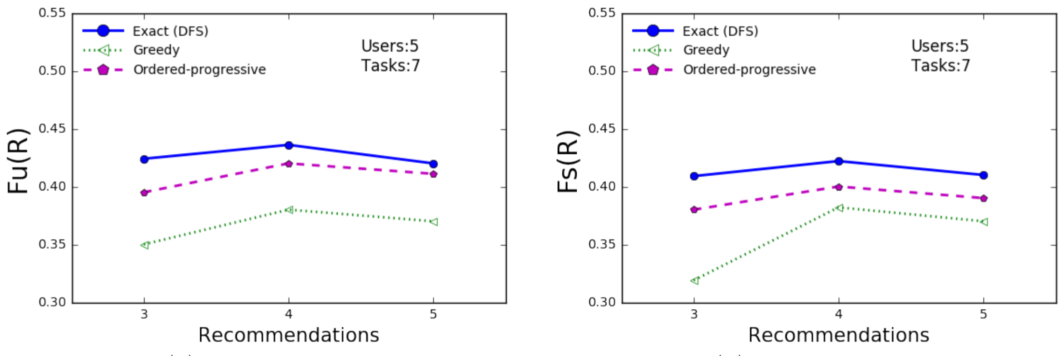


(b) Task Utility

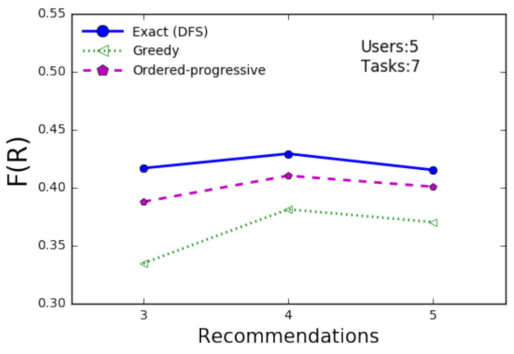


(c) System Utility

Figure 3.9: Utility measures as a function of the users.



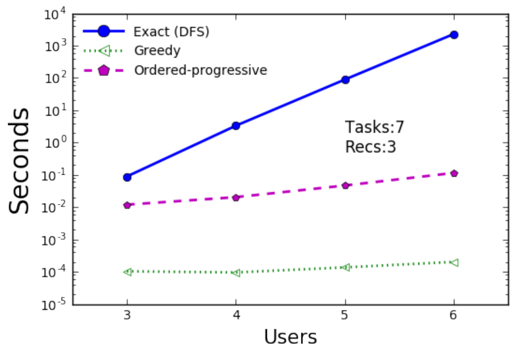
(a) User Utility (b) Task Utility



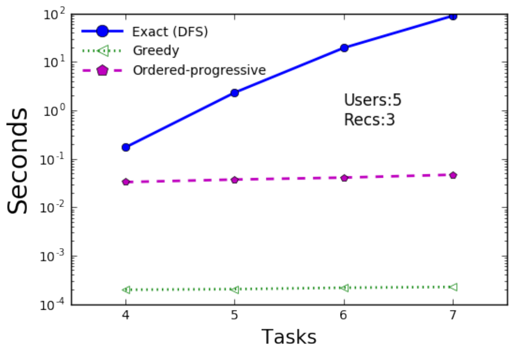
(c) System Utility

Figure 3.10: Utility measures as a function of the recommendations.

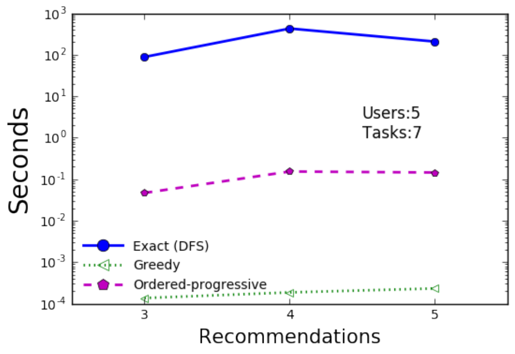
(Mean Squared Error = 0.521 , Relative Error = 0.477) and benefit (Mean Squared Error = 0.274, Relative Error = 0.346). As a result, the recommendations obtained by both methods were similar to the results obtained using the real data. On the other hand, the hybrid/ordered-progressive algorithm outperformed the greedy algorithm both for the user and the task utility. Figure 3.16(b) shows that the greedy approach had a higher standard deviation than hybrid/ordered-progressive. This could be because the algorithm would sometimes recommend the same task to several users, making some task probabilities of being performed close to 1, while neglecting other tasks and making their probability of being performed 0 or close to 0. This is also



(a) As a function of users



(b) As a function of tasks

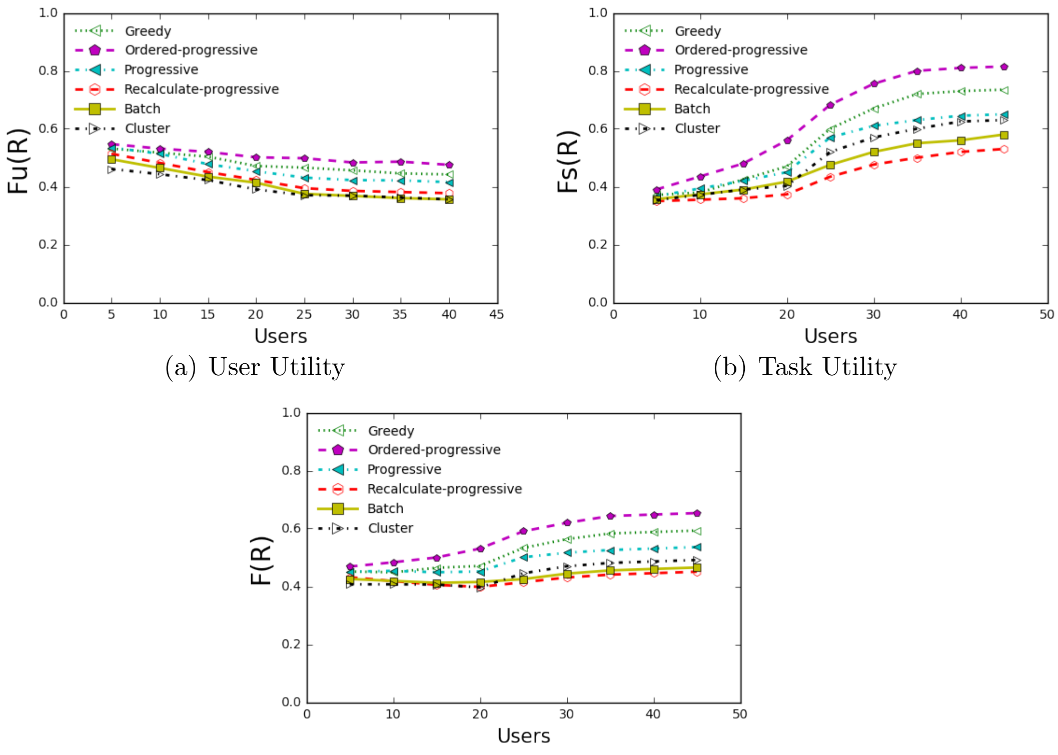


(c) As a function of number of recommendations

Figure 3.11: Comparison of the running time for the different methods

reflected in a lower user utility in the greedy algorithm (lower $P(u_i \rightarrow s_j)$). Hybrid/ordered-progressive effectively recommends other tasks to users because it considers the dependent probability of performing a task. This is reflected not only in a higher user and task utility, but also in a lower standard deviation for the task utility. Table 3.1 shows a summary of the data from the box plots.

For performing the analysis with the six different datetimes, we considered the utility in all of the dates. Some datetimes had less users and tasks (e.g. Thursday at 4:10 a.m. had 388 users and 392 tasks) while others had more



(a) User Utility

(b) Task Utility

(c) System Utility

Figure 3.12: Comparison of the results of the simulations as a function of the number of users

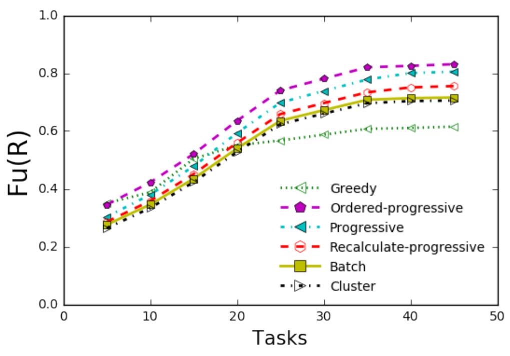
(e.g. Monday at 7:00 p.m. had 3584 users and 2937 tasks). We added the user utility of all the users and divided over the total number of users (13,278). We likewise obtained the task utility of all the tasks (the total number of tasks was 11,353). We obtained a weighted standard deviation, which was proportional to the users and tasks in each datetime (e.g. the weight for the standard deviation in the user utility for Monday at 7:00 p.m. was 3584/13278). Table 3.2 shows the results for the five datetimes, and figure 3.17 shows the corresponding box plot.

Data	Method	F(R)	Utility	n	mean	std	Q1	Q3
Real	Greedy	0.382	User	3151	0.279	0.093	0.211	0.338
			Task	2983	0.485	0.226	0.306	0.666
	Ord-Prog	0.487	User	3151	0.369	0.122	0.278	0.453
			Task	2983	0.606	0.083	0.548	0.663
Predicted	Greedy	0.347	User	3151	0.256	0.089	0.194	0.315
			Task	2685	0.439	0.235	0.25	0.624
	Ord-Prog	0.454	User	3151	0.359	0.109	0.273	0.432
			Task	2685	0.549	0.075	0.543	0.644

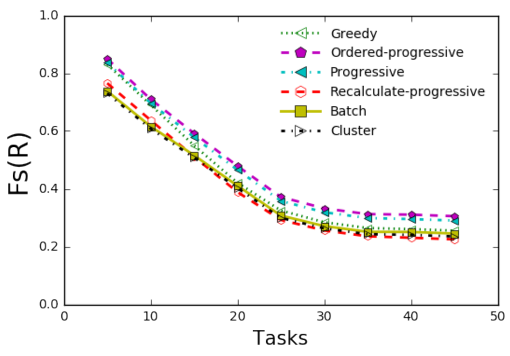
Table 3.1: Summary of the data from the box plots

Data	Method	F(R)	Utility	n	mean	std	Q1	Q3
Real	Greedy	0.393	User	15968	0.348	0.114	0.258	0.418
			Task	13981	0.437	0.279	0.268	0.684
	Ord-Prog	0.6175	User	15968	0.491	0.124	0.404	0.577
			Task	13981	0.692	0.089	0.555	0.684
Predicted	Greedy	0.402	User	15968	0.352	0.089	0.267	0.422
			Task	12418	0.478	0.266	0.268	0.684
	Ord-Prog	0.53	User	15968	0.509	0.144	0.407	0.61
			Task	12418	0.737	0.103	0.578	0.717

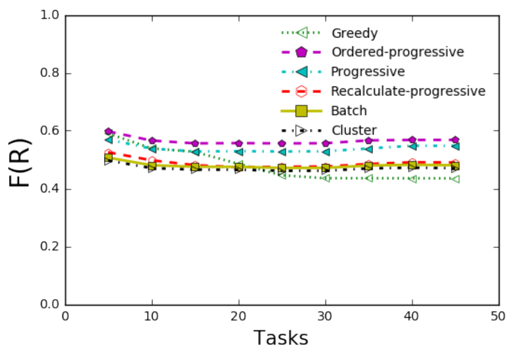
Table 3.2: Summary of the data from the box plots



(a) User Utility

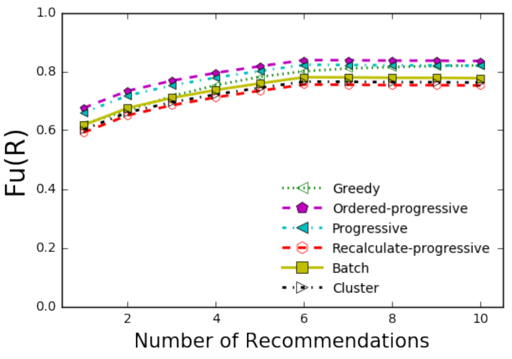


(b) Task Utility

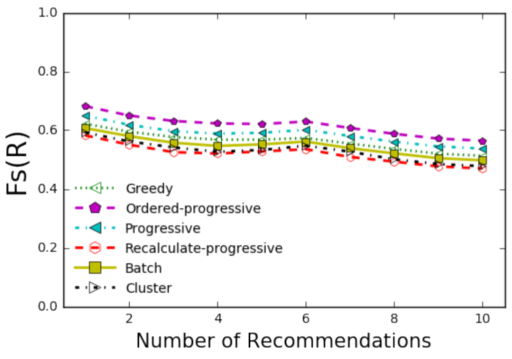


(c) System Utility

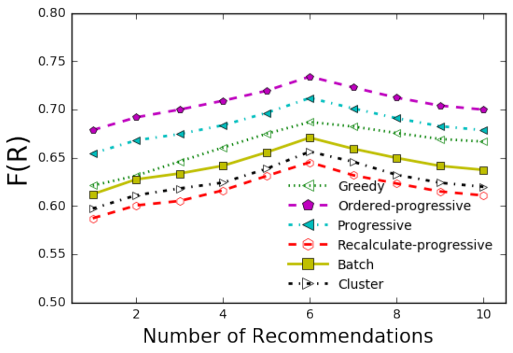
Figure 3.13: Comparison of the results of the simulations as a function of the number of tasks



(a) User Utility

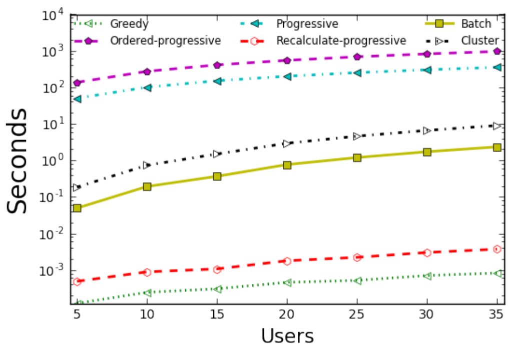


(b) Task Utility

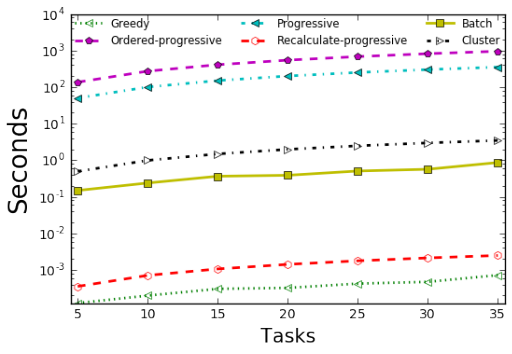


(c) System Utility

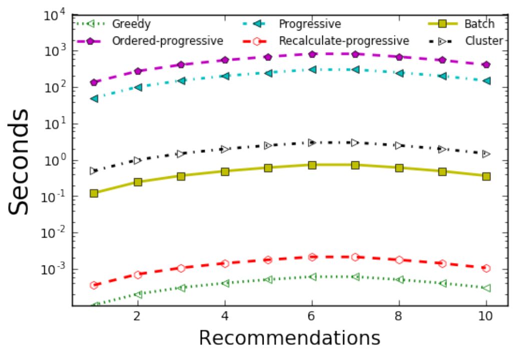
Figure 3.14: Comparison of the results of the simulations as a function of the number of recommendations



(a) Running time as a function of users

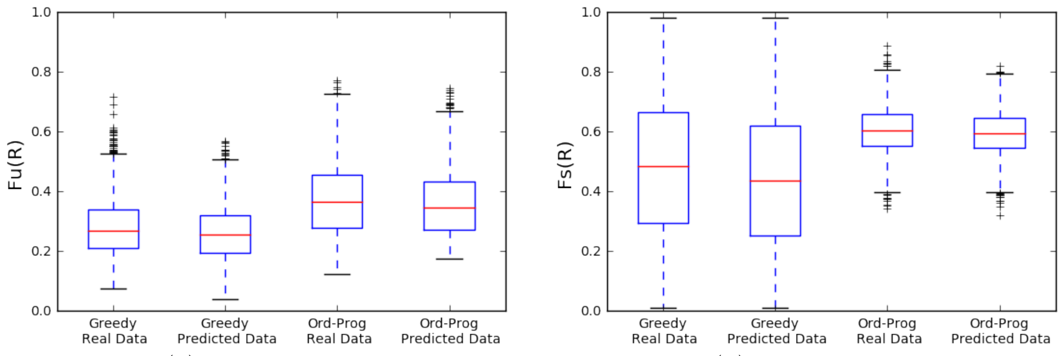


(b) Running time as a function of tasks



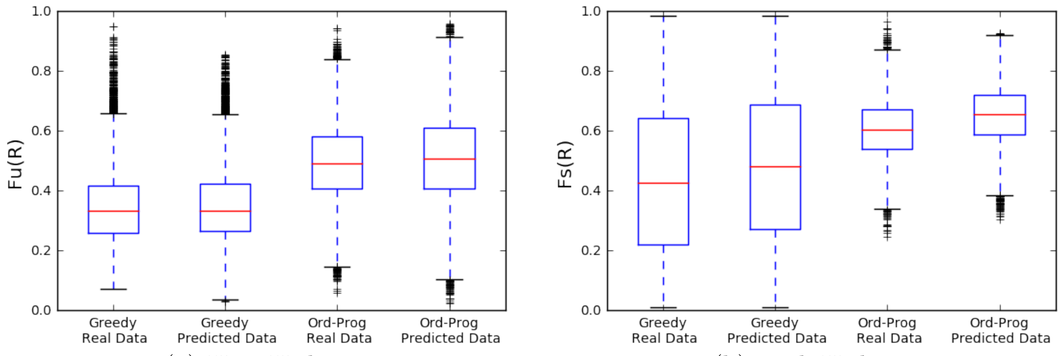
(c) Running time as a function of number of recommendations

Figure 3.15: Running time of simulations



(a) User Utility (b) Task Utility

Figure 3.16: Comparison of the models run with real world data and with the data predicted from the tensor factorization



(a) User Utility (b) Task Utility

Figure 3.17: Comparison of the models run with real world data and with the data predicted from the tensor factorization. Data from 5 different data points

Chapter 4

Location and Time-aware Truth-Inference with Bayesian Filtering

4.1 Problem Formulation

Consider the spatial crowdsourcing scenario, in which a set of n_{users} different users join the task of reporting specific events at n_{locs} different locations and at n_{times} different and consecutive time slices. We denote the set of these users as $U = \{u_i | i = 1, ..., n_{users}\}$, the set of locations of interest as $L = \{l_j | j = 1, ..., n_{locs}\}$ and the set of relevant times as $T = \{t_k | k = 1, ..., n_{times}\}$. At any time $t_k \in T$, an event in location $l_j \in L$ could be true (denoted by '1') or false (denoted by '0'). Let Z be the set of events, where each element $z_{jk} = 1$ iff the event at time t_k and location l_j is true. We present our problem and solutions for events with a binary state here, but we note that they can be extended to events with multiple states.

For example, a set of $n_{users} = 5$ users could be asked to report if there is high traffic at $n_{locs} = 10$ different street intersections during the $n_{times} = 24$ hours of a certain day, or they could be asked to report if gas is lower than

\$2.50 at $n_{locs} = 3$ different gas stations for $n_{times} = 3$ days. The price at location 1 for the three days could be \$2.40, \$2.45, and \$2.58 respectively (Figure 1.2(a)), so the true states of the event are $z_{11} = 1, z_{12} = 1$, and $z_{13} = 0$ for location 1.

The users would simply report '1' if the event is true. We assume there is a default state (e.g. gas price is \$2.50 or more, light traffic) and consider therefore that the reports could be of two kinds: 1) *Positive report*, where the user reports the event as true (denoted as '1'); and 2) *Missing report* where the user did not send a report (denoted as '0'). Missing reports could occur in a variety of circumstances, such as the user not being at the specific place and time, lack of participation from the user, or the event being false. In other words, a missing report could mean both absence of report or report of a negative state. To distinguish between these scenarios, we take into consideration both the probability that a user was at the specific time and place of the event and the reliability traits of the user, although these parameters are not known beforehand. Our main goal is to determine the label of z_{jk} as true or false for all of the relevant locations and times.

Figure 1.2 shows an example of a spatio-temporal crowdsourced task. Users are moving and report if an event is true at different locations of interest. The event could be gas price below \$2.50. Figure 1.2(b) shows an example of received reports throughout three different days. Based on these reports and without knowledge of the users' true locations, the goal is to infer the true label of the events over time (figure 1.2(c)). We consider different types of users: 1) trustworthy users (e.g. user 1 and 3) who report the events true when they are true; 2) passive users (e.g. user 2) who never send reports despite being in the correct location and time; 3) untrustworthy users (e.g. user 4) who send the wrong reports; and 4) malicious users (e.g. user 5) that send reports about events they are not observing.

Our assumption of a default state is motivated by a variety of applications

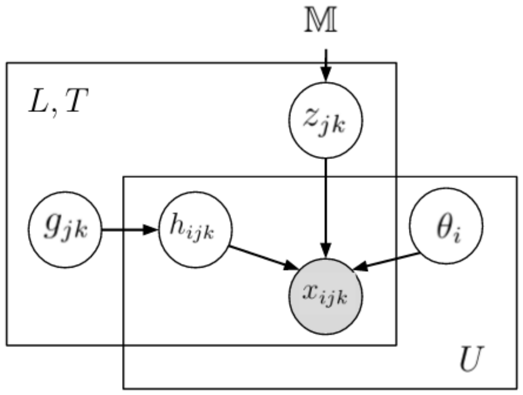


Figure 4.1: Graphical Model showing the dependency of the variables. U, L, and T are the set of Users, Locations, and Time steps, respectively

where users normally do not report a negative state. For example, users do not normally send a report when there is no traffic, since that is the default state. Other examples with a default state include sending a report when there is graffiti on a wall, reporting potholes or trash on the street, locations with free Wi-Fi, and accidents on a road.

Our model can be extended to include multiple states (e.g. report on low, medium, or high traffic), discretize continuous variables (e.g. the price of gas is either below \$1.50, between \$1.50 and \$3, or higher than \$3), or have a "negative" report (e.g. there is no accident at this location). We further discuss these scenarios in section 5.3.

4.1.1 Proposed Model

In this section we present our proposed graphical model that specifies the dependence at a given time point between the report of a user, the true state of the event at a given location, and other factors. Figure 4.1 shows our graphical model and Table 4.1 summarizes its notations. The model consists of the following elements:

Table 4.1: Notation for the Graphical Model

Symbol	Meaning
n_{users}, n_{locs}, n_{times}	Number of users, event locations, and time slices, respectively
U, L T	Set of Users, Locations, and Time slices, respectively
$\mathbb{M}$	Event model for the events Z
z_{jk}	Binary variable with the state of the event at location l_j and time t_k
x_{ijk}	Binary variable with the report of user u_i at location l_j at time t_k
h_{ijk}	Binary variable that indicates if user u_i was at location l_j at time t_k
g_{jk}	Popularity of location l_j at time t_k
θ_i	Reliability model for user u_i

Reports from the users.

The reports from the users are collected in the form of a three dimensional binary variable $X = \{x_{ijk}\}$, where $x_{ijk} = 1$ iff user u_i reported the event at location l_j and time t_k as true. Since a user u_i cannot be in two locations at a same time t_k, $x_{i\mathbf{j}k} = 1$ for some $\mathbf{j} \in \{1, ..., n_{locs}\} \Rightarrow x_{ijk} = 0 \quad \forall j \neq \mathbf{j}$.

On the other hand, $x_{ijk} = 0$ could be because the user u_i was not at location l_j at time t_k, the event z_{jk} was not true, or because the user decided not to participate. Our method establishes a probability model to explain the different possibilities, which are detailed in Section 4.1.1.

Label of the event.

The label of the event at location l_j and time t_k is denoted as z_{jk} and it is a latent binary variable and one we ultimately seek to estimate based on the reports from the users. We assume that there is an *event model* $\mathbb{M}$ that models the spatio-temporal correlations between the event states at consecutive time slices and determines the value of z_{jk}. In Section 4.2.1, we will assume that $\mathbb{M}$ is not known and present a method based on Bayesian Estimation, and then we present an enhanced method that explicitly specifies and estimates $\mathbb{M}$ in Section 4.2.2.

Popularity of the locations.

One of the factors that influences the reports from the users is if the user is actually at the specified location. Establishing the popularity of each place will be important to establish the probability that a user was actually at a determined location at a certain time. The popularity of a place is defined as the probability that a user chosen at random will be at that place at any time. The popularity usually follows a power law distribution, so that roughly 20% of the places have 80% of the popularity [64]. Let g_{jk} be the

popularity of location l_j at time t_k. For certain applications, and depending on the time periods, it is reasonable to assume the popularity of a location is independent of time. In such cases, we can drop the time index and $g_{jk} = g_j$.
User's location indicator. We use a binary variable $H = \{h_{ijk}\}$ to indicate if user u_i was at location l_j at time t_k. This variable is determined by the popularity of the locations g_{jk}. Since we assume that we are not tracking the users at all times, this variable is hidden from us, but our method is capable of determining a probabilistic approximation of H that will ultimately be useful to establish the reliability model for the users. Similarly to the reports, $h_{i\mathbf{j}k} = 1$ for some $\mathbf{j} \Rightarrow h_{ijk} = 0 \ \ \forall j \neq \mathbf{j}$, since a user cannot be in two locations at a same time. On the other hand, $h_{ijk} = 0$ directly implies that the user u_i was not at location l_j at time t_k. This is different from the reports, where $x_{ijk} = 0$ could be for a series of reasons. Although some mobile apps send GPS location together with the report, we are assuming that this data is not specifically sent. On the other hand, if GPS data is available, we could incorporate this information.

User's reliability model.

An important factor that determines the reports from the users is their reliability traits. We assume that a user is going to report an event with the same probability regardless of the location they are in or the time slice. Therefore, the reliability model does not vary as a function of the locations L or the time slices T. We model the user's reliability in the following way. For each user $u_i \in U$, for all $l_j \in L$, and $t_k \in T$,

$$\alpha_i = P(x_{ijk} = 1 | h_{ijk} = 1, z_{jk} = 1) \tag{4.1}$$

$$\beta_i = P(x_{ijk} = 1 | h_{ijk} = 1, z_{jk} = 0) \tag{4.2}$$

$$\gamma_i = P(x_{ijk} = 1 | h_{ijk} = 0) \tag{4.3}$$

The probability that a user u_i will report an event as true given that the user is in the correct location and time and that the event is actually true is represented by α_i. The probability that user u_i will report an event as true given that u_i is in the correct location and time, but the event is false is represented by β_i. Finally, γ_i is the probability that a user reports an event as true, given that the user was not at the specified time and location and regardless of whether the event was true or not.

We discuss several kinds of typical users which can be modeled based on different values for α, β, and γ. A user with γ_i close to 1 would be a malicious user, since the user is reporting an event that is not being observed. In general, we expect that the users have no incentive to be malicious and that γ_i is close to zero. Assuming γ_i close to zero, a trustworthy user u_i would have α_i close to one and a low β_i, and would send reports iff u_i observes the event as true. An aggressive user would have both α_i and β_i close to one, and would send reports if the event is true and sometimes also when it is false. This could be due to a misinterpretation of the event. A passive user would have low α_i and β_i, and would not send reports often, regardless of the label of the event. Finally, an untrustworthy user would have low α_i and high β_i, and would report events as true when they are false, and not report events when they are true.

For ease of notation, let $\theta_i = (\alpha_i, \beta_i, \gamma_i)$ and $\Theta = \{\theta_i | i = 1, ..., n_{users}\}$. Variable Θ completely determines the reliability traits of the users, and is able to describe their trustworthiness, willingness to cooperate, and maliciousness. Table 4.2 is a summary of the probabilities of X given Z, H, and Θ under all circumstances.

The model presented here allows a **probabilistic explanation for missing reports** in the following way. A report could be missing ($x_{ijk} = 0$) if the user was not at the correct time and place, if the state of the event was the default (i.e. not true), or if the user preferred not to participate. Now

Table 4.2: Probabilities of reports given H, Z, and θ. For example, $P(x_{ijk} = 1|h_{ijk} = 1, z_{jk} = 1) = \alpha_i$

$h = 1$		x_{ijk}	
		0	1
z_{jk}	0	$(1 - \beta_i)$	β_i
	1	$(1 - \alpha_i)$	α_i

$h = 0$		x_{ijk}	
		0	1
z_{jk}	0	$(1 - \gamma_i)$	γ_i
	1	$(1 - \gamma_i)$	γ_i

that we have defined our model, we calculate the probability of each of these cases, and determine that:

$$\begin{aligned}
P(x_{ijk} = 0) =& P(h_{ijk} = 0)(1 - \gamma_i) + \\
& P(h_{ijk} = 1)(1 - P(z_{jk} = 1))(1 - \beta_i) + \\
& P(h_{ijk} = 1)P(z_{jk} = 1)(1 - \alpha_i) \\
=& (1 - \hat{h}_{ijk})(1 - \gamma_i) + \\
& \hat{h}_{ijk}(1 - \hat{z}_{jk})(1 - \beta_i) + \\
& \hat{h}_{ijk}\hat{z}_{jk}(1 - \alpha_i)
\end{aligned}$$

Where $\hat{z}_{jk}$ is an estimation of the true value of z_{jk} and is defined as $\hat{z}_{jk} = P(z_{ij} = 1)$. Analogously, $\hat{h}_{ijk} = P(h_{ijk} = 1)$. The above equation clearly separates the reasons why a report might be missing into three different terms. A similar analysis could be used to determine the reasons for a positive report.

4.2 Truth-Inference Algorithm

We first discuss the Bayesian Estimation part of the Truth Inference algorithm, which infers the labels of the events without using the event model, and then we improve upon it with the Kalman Estimation, which explicitly models the event with a state-space model $\mathbb{M}$.

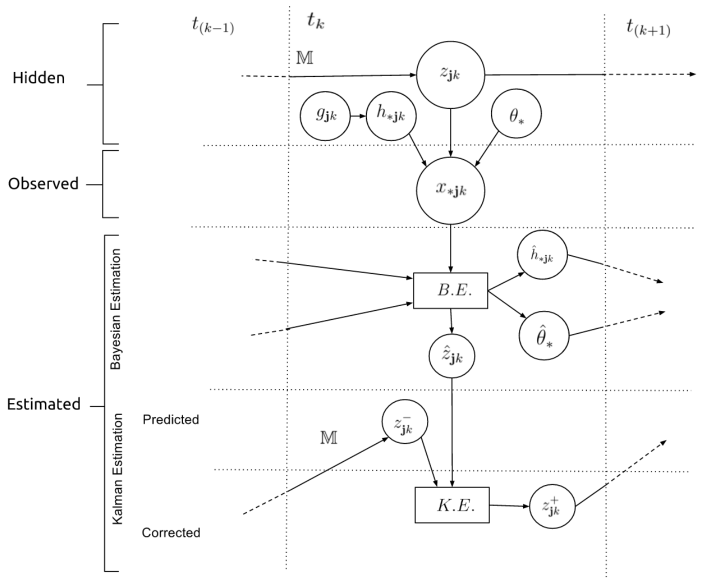


Figure 4.2: Bayesian Estimation and Kalman Estimation

4.2.1 Bayesian Estimation

We discuss how to obtain the latent variables Z, H, and Θ through a continuous approximation in a recursive way based on the reports X of the users.

Setting the initial values.

We start the algorithm by setting initial values for the latent variable Θ. In general, any random number would work, but we can make some assumptions that will speed up the convergence of the method. We assume that the users do not have any incentives to report an event as true when they are not in the location and time of interest, so we set $\hat{\gamma}_i$ close to zero.[1] On the other hand, a value of β_i close to 1 would indicate that the user tends to misinterpret the label of the event. In the case of clear-cut events (e.g. the price of gas

[1] we use the conventional "hat" notation to mean an estimated value

less than \$2.50) there is not much room for misinterpretation, so we could assign a value of $\hat{\beta}_i$ close to zero. A value of α_i close to 1 could be used for applications where there is an incentive for cooperating. The application to report traffic in Waze, for example, has the incentive of helping others avoid traffic jams and a sense of community. Depending on the application at hand, we could assign values of $\hat{\alpha}_i$ close to 1, where $\hat{\alpha}_i = 1$ implies perfect participation and trustworthiness.

Since we are assuming that there is no incentive to report from a far-away location, we set $\hat{h}_{\mathbf{ijk}} = 1$ whenever $x_{\mathbf{ijk}} = 1$. In such case, we also set $\hat{h}_{\mathbf{ijk}} = 0\ \forall j \neq \mathbf{j}$ since a user can only be at one location at one time. The rest of the values of $\hat{H}$, where there were no reports at location l_j at time t_k are estimated as follows. Since $\hat{h}_{ijk}$ is interpreted as the continuous variable that determines the probability that user u_i was at location l_j at time t_k, and g_j is the probability that a randomly chosen participant will visit the location at least once in $t = 1, ..., n_{times}$, it follows that for a particular user u_i and location l_j, $\prod_{k=1}^{n_{times}}(1 - h_{ijk})$ is the probability that user u_i will not visit l_j in any time period, which is equal to $(1 - g_j)$. If we assume that all time periods are equally popular for the same location, then $h_{ijk} = h_{ij1}$ for all k, and $\prod_{k=1}^{n_{times}}(1 - h_{ijk}) = \prod_{k=1}^{n_{times}}(1 - h_{ij1}) = (1 - h_{ij1})^{n_{times}} = (1 - g_j)$.We then find that the value of $h_{ijk} = 1 - (1 - g_j)^{1/n_{times}}$ for all k where $x_{ijk} = 0$.

For the initial values of $\hat{Z}$, we set $\hat{z}_{jk} = P(z_{jk} = 1) = (1/\sum_{u_i \in U} x_{ijk}) \sum_{u_i \in U} x_{ijk}\hat{h}_{ijk}$. This first approximation to $\hat{Z}$ is a "weighted majority voting" approach where the weights are determined by the probability of the users being there. However, it does not consider all of the reasons previously discussed for missing reports.

Updating the variables

Once we have initial values for the latent variables, we continue to update their values with the available reports by using equations of total probability and Bayes' theorem recursively given our dependency graph in Figure 4.1.

Updating $\hat{H}$. In the case of H, for each time t_k, location l_j and user u_i, we update the value of $\hat{h}_{ijk}$ using the available reports X and the equation for total probability:

$$\begin{aligned} P(h_{ijk}=1|X) =& P(h_{ijk}=1|X, z_{jk}=1)P(z_{jk}=1)+ \\ & P(h_{ijk}=1|X, z_{jk}=0)P(z_{jk}=0) \end{aligned} \tag{4.4}$$

Taking the first term when the true label of the event is 1 ($z_{jk}=1$), and assuming that the reported value is 1 ($x_{ijk}=1$), we can use Bayes' theorem and obtain that:

$$\begin{aligned} & P(h_{ijk}=1|x_{ijk}=1, z_{jk}=1) \\ =& \frac{P(x_{ijk}=1, z_{jk}=1|h_{ijk}=1) \times P(h_{ijk}=1)}{P(x_{ijk}=1, z_{jk}=1)} \\ =& \frac{P(x_{ijk}=1|h_{ijk}=1, z_{jk}=1)P(z_{jk}=1|h_{ijk}=1)P(h_{ijk}=1)}{P(x_{ijk}=1|z_{jk}=1)P(z_{jk}=1)} \\ =& \frac{\hat{\alpha}_i \times P(z_{jk}=1) \times \hat{h}_{ijk}}{(\hat{\alpha}_i \times \hat{h}_{ijk} + \gamma_i(1-\hat{h}_{ijk})) \times P(z_{jk}=1)} = \frac{\hat{\alpha}_i \hat{h}_{ijk}}{\hat{\alpha}_i \hat{h}_{ijk} + \hat{\gamma}_i(1-\hat{h}_{ijk})} \end{aligned} \tag{4.5}$$

Likewise, if we now assume that $x_{ijk}=0$, we analogously determine that:

$$P(h_{ijk}=1|x_{ijk}=0, z_{jk}=1) = \frac{(1-\hat{\alpha}_i)\hat{h}_{ijk}}{(1-\hat{\alpha}_i)\hat{h}_{ijk} + (1-\hat{\gamma}_i)(1-\hat{h}_{ijk})} \tag{4.6}$$

For the second term, where we have $z_{jk}=0$, and assuming $x_{ijk}=1$:

$$P(h_{ijk}=1|x_{ijk}=1, z_{jk}=0) = \frac{\hat{\beta}_i \hat{h}_{ijk}}{\hat{\beta}_i \hat{h}_{ijk} + \hat{\gamma}_i(1-\hat{h}_{ijk})} \tag{4.7}$$

Finally, when $z_{jk}=0$ and assuming $x_{ijk}=0$, then

$$P(h_{ijk}=1|x_{ijk}=0, z_{jk}=0) = \frac{(1-\hat{\beta}_i)\hat{h}_{ijk}}{(1-\hat{\beta}_i)\hat{h}_{ijk} + (1-\hat{\gamma}_i)(1-\hat{h}_{ijk})} \tag{4.8}$$

Therefore, if a report $x_{ijk} = 1$, then we use equations (4.4),(4.5), and (4.7) to update H, and equations (4.4), (4.6), and (4.8) if $x_{ijk} = 0$. They are shown respectively as follows:

$$\begin{aligned} P(h_{ijk} = 1 | x_{ijk} = 1) = & \frac{\hat{\alpha}_i \hat{h}_{ijk} \hat{z}_{jk}}{\hat{\alpha}_i \hat{h}_{ijk} + \hat{\gamma}_i (1 - \hat{h}_{ijk})} + \\ & \frac{\hat{\beta}_i \hat{h}_{ijk} (1 - \hat{z}_{jk})}{\hat{\beta}_i \hat{h}_{ijk} + \hat{\gamma}_i (1 - \hat{h}_{ijk})} \end{aligned} \tag{4.9}$$

$$\begin{aligned} P(h_{ijk} = 1 | x_{ijk} = 0) = & \frac{(1 - \hat{\alpha}_i) \hat{h}_{ijk} \hat{z}_{jk}}{(1 - \hat{\alpha}_i) \hat{h}_{ijk} + \hat{\gamma}_i (1 - \hat{h}_{ijk})} + \\ & \frac{(1 - \hat{\beta}_i) \hat{h}_{ijk} (1 - \hat{z}_{jk})}{(1 - \hat{\beta}_i) \hat{h}_{ijk} + (1 - \hat{\gamma}_i)(1 - \hat{h}_{ijk})} \end{aligned} \tag{4.10}$$

Updating $\hat{Z}$. Updating the values of $\hat{Z}$ follows a very similar procedure, where we will be updating its values using the values of the recently calculated $\hat{H}$ and our observations of X. We follow an analogous reasoning as for equation (4.4):

$$\begin{aligned} P(z_{jk} = 1 | X) = & P(z_{jk} = 1 | X, h_{ijk} = 1) P(h_{ijk} = 1) + \\ & P(z_{jk} = 1 | X, h_{ijk} = 0) P(h_{ijk} = 0) \end{aligned}$$

Analogously to equations (4.5-4.8), we construct the equations depending on the observed reports:

$$\begin{aligned} P(z_{jk} = 1 | x_{ijk} = 1) = & \frac{\hat{\alpha}_i \hat{z}_{jk} \hat{h}_{ijk}}{\hat{\alpha}_i \hat{z}_{jk} + \hat{\beta}_i (1 - \hat{z}_{jk})} + \\ & \frac{\hat{\gamma}_i \hat{z}_{jk} (1 - \hat{h}_{ijk})}{\hat{\gamma}_i \hat{z}_{jk} + (1 - \hat{\gamma}_i)(1 - \hat{z}_{jk})} \end{aligned} \tag{4.11}$$

$$\begin{aligned} P(z_{jk} = 1 | x_{ijk} = 0) = & \frac{(1 - \hat{\alpha}_i) \hat{z}_{jk} \hat{h}_{ijk}}{(1 - \hat{\alpha}_i) \hat{z}_{jk} + \hat{\beta}_i (1 - \hat{z}_{jk})} + \\ & \frac{(1 - \hat{\gamma}_i) \hat{z}_{jk} (1 - \hat{h}_{jk})}{(1 - \hat{\gamma}_i) \hat{z}_{jk} + \hat{\gamma}_i (1 - \hat{z}_{jk})} \end{aligned} \tag{4.12}$$

The final estimation for z_{jk} is an aggregation over all the reports from the users.

Updating $\hat{\Theta}$. Finally, for updating $\hat{\Theta}$, we follow the definitions of α_i, β_i, and γ_i from section 4.1.1 and applying Bayes theorem, arrive at the following update equations:

$$\hat{\alpha}_i = \frac{\hat{z}_{jk}\hat{h}_{ijk}}{\hat{z}_{jk}\hat{h}_{ijk} + (1-\hat{z}_{jk})(1-\hat{h}_{ijk})} \tag{4.13}$$

$$\hat{\beta}_i = \frac{(1-\hat{z}_{jk})\hat{h}_{ijk}}{(1-\hat{z}_{jk})\hat{h}_{ijk} + \hat{z}_{jk}(1-\hat{h}_{ijk})} \tag{4.14}$$

$$\hat{\gamma}_i = \frac{\hat{z}_{jk}(1-\hat{h}_{ijk})}{\hat{z}_{jk}(1-\hat{h}_{ijk}) + (1-\hat{z}_{jk})\hat{h}_{ijk}} + \frac{(1-\hat{z}_{jk})(1-\hat{h}_{ijk})}{(1-\hat{z}_{jk})(1-\hat{h}_{ijk}) + (1-\hat{z}_{jk})\hat{h}_{ijk}} \tag{4.15}$$

This is essentially an expectation maximization algorithm, since we are updating the variables in the expectation phase and maximizing the value of $P(X|\hat{H}, \hat{\Theta}, \hat{Z})$.

Convergence.

We iteratively continue updating the variables $\hat{H}, \hat{Z}$, and $\hat{\Theta}$ until convergence, which is determined by the relative change of the variables $\hat{Z}$ between one iteration and the next. In other words, we stop if at some iteration r, $||\hat{Z}^{(r-1)} - \hat{Z}^{(r)}||/||\hat{Z}^{(r-1)}|| \leq \epsilon$ for some tolerance ϵ. The convergence of $\hat{Z}$ considers all the locations and all the times. The resulting $\hat{Z}$ has values in the interval $[0, 1]$, which are interpreted as the probability of an event to be true. If we need to decide a True/False label for the event, we set a threshold τ (say $\tau = \frac{1}{2}$) and label the event at location l_j at time t_k as true iff $\hat{z}_{jk} \geq \tau$. Algorithm 5 summarizes the method.

Algorithm 5 Truth Inference from Spatio-Temporal reports using Bayesian Estimation (BE)

```
Input: Spatio-temporal reports from the users: $X$
Output: 1) Labels of the events: $\hat{Z}$;
2) Reliability Model $\hat{\Theta}$
 1: Initialize $\hat{\Theta}$ with random numbers. It is useful to incorporate beliefs to
    speed up convergence.
 2: Initialize $\hat{H}$ in the following way:
        - For all i,j,k such that $x_{\mathbf{ijk}} = 1$, set $\hat{h}_{\mathbf{ijk}} = 1$ and $\hat{h}_{\mathbf{i}j\mathbf{k}} = 0\ \forall j \neq \mathbf{j}$
        - For all i,j,k such that $x_{\mathbf{ijk}} = 0$, set $\hat{h}_{\mathbf{i}j\mathbf{k}}$ according to the popularity
          of $l_j$ at time $t_k$
 3: Initialize $\hat{Z}$ by setting $\hat{z}_{jk} = \frac{1}{|U|}\sum_{u_i \in U} x_{ijk}$ for each $l_j \in L, t_k \in T$
 4: Set $r = 1$, $\hat{H} = \hat{H}^{(0)}$, $\hat{Z} = \hat{Z}^{(0)}$, $\hat{\Theta} = \hat{\Theta}^{(0)}$
 5: while has not converged do
 6:     Update $\hat{H}^{(r)}$ with equations (4.9-4.10)
 7:     Update $\hat{Z}^{(r)}$ with equations (4.11-4.12)
 8:     Update $\hat{\Theta}^{(r)}$ with equations (4.13, 4.14, 4.15)
 9:     Check for convergence using $\hat{Z}^{(r)}$ and $\hat{Z}^{(r-1)}$
10:     $r = r + 1$
11: end while
12: For all $l_j \in L$, $t_k \in T$, if $\hat{z}_{jk} \geq \tau$ for some threshold $\tau$, label event at
    location $l_j$, time $t_k$ as true. Otherwise, label it as false.
```

4.2.2 Kalman Estimation

The recursive Bayesian method obtains the probability that an event is true given the reports from the crowdsourced data. When we consider the mathematical or probabilistic model for the behaviour of the events (event model), we can obtain a better approximation to Z. We use the Kalman filter, which is a recursive algorithm that uses data from the previous state of the variables (i.e. the previous time-step, or t_{k-1}) together with observations from the current state (at t_k) and makes a prediction based on the underlying model to estimate the unknown or latent variables [42].

Event Model. The event model is used in the *Prediction phase* of the Kalman Estimation. We assume that there exists an event model $\mathbb{M}$ that determines the values of the labels of the events z_{jk}. For example, the value of z_{jk} could be determined by a Markov model if its value depended only on the value of $z_{j(k-1)}$:

$$z_{jk} = \mathbb{M} z_{j(k-1)} \tag{4.16}$$

It could also be determined by time series models if the value of z_{jk} depended on $z_{j(k-1)}, z_{j(k-2)}, ..., z_{j1}$. The value of z_{jk} could also be determined by the values of $z_{\mathbf{j}k}, \mathbf{j} \neq j$ if there is a correlation among the locations. For example, the traffic at location $l_{\mathbf{j}}$ could be determined by the surrounding locations since the traffic could start spreading to nearby areas. In general, the event model $\mathbb{M}$ for the values of Z is not known.

In such cases, we can use the observed data from the previous time-steps to train the model $\mathbb{M}$. To continue our example with gas prices, assume we do not have previous data for the prices, and we only have our estimation based on observations at t_1. A prediction for t_2 could be made by assuming $\mathbb{M} = \mathbb{I}$, so that the gas prices at t_2 would be the same as in t_1. If we observe any changes at t_2 after making an estimation and a correction, then we start adjusting the model. Since the gas prices could be modeled as a time series, we can use an autoregressive moving average to make our next prediction at t_3 [65]. The model would be adjusted accordingly as more observations are made.

Initialization of the recursive implementation. In our recursive implementation, for each location j, and at t_1, the approximation obtained from the Bayesian estimation $\hat{z}_{j1}$ is used to predict the value of $\hat{z}_{j2}$ using the underlying model $\mathbb{M}$ (prediction phase). Let z_{j2}^{-} be the *prior* or predicted value.[2] On the other hand, at time t_2 we have our Bayesian estimation $\hat{z}_{j2}$. Using these two values z_{j2}^{-} and $\hat{z}_{j2}$, we use the Kalman estimation to obtain a corrected estimation of z_{j2} (correction phase). The next subsection details the prediction and correction phase. Let z_{j2}^{+} be the *posterior*[3] or the value approximated by the Kalman estimation. We can now apply the underlying mathematical model to the posterior value to obtain a prior for the next time step. With this initialization, this recursive algorithm can be applied at $t_2, ..., t_{n_{times}}$. We now explain in more detail the two phases of the Kalman Estimation part of the algorithm.

Prediction Phase. In the prediction phase at time t_k, we use the underlying event model $\mathbb{M}$ to make a prediction for time t_{k+1}.

In general, if we consider T to be the training data, either obtained offline, online, or both, and $z_{j(k-1)}^{+}$ the corrected approximation for the label at a previous time-step, then the a priori prediction for time t_k is

$$z_{jk}^{-} = \mathbb{M}(T, z_{j(k-1)}^{+}) \tag{4.17}$$

Correction Phase. In this phase, we use the prediction from the previous phase together with the estimated variable from the observed reports at the current time-step to update the approximation to the true label of the event.

[2]We use the conventional $^{-}$ notation to denote the a priori prediction

[3]Likewise, the $^{+}$ notation denotes the posterior estimation

If z^-_{jk} is the a priori prediction, and $\hat{z}_{jk}$ is the approximation obtained from the observations at location l_j and time t_k, then

$$z^+_{jk} = z^-_{jk} + K_k \times (\hat{z}_{jk} - z^-_{jk}) \tag{4.18}$$

is the posterior or corrected approximation to z_{jk}, where K_k is the *Kalman gain* at time t_k. Details on the derivation of the Kalman gain can be found in the seminal work by R.E. Kalman [42]. In the unidimensional case, the posterior is a convex linear combination of the prior and the estimated values. If $K_k = 0$ then $z^+_{jk} = z^-_{jk}$ and $K_k = 1$ implies $z^+_{jk} = \hat{z}_{jk}$, so the Kalman gain is a term that "corrects" the prediction using the estimated value from the observed data. We also consider the multidimensional case when there is a temporal-spatial correlation of the events. Such cases occur, for example, if high traffic in one location correlates with high traffic in another location.

Algorithm 6 Truth Inference from Spatio-Temporal reports using Bayesian Estimation and Kalman Estimation (BE+KE)

```
Input: Spatio-temporal reports from the users $X$; Event model $\mathbb{M}$
Output: Labels of the events: $Z^+$;
1: Apply algorithm 5 (BE) to $x_{ij1}$ to obtain $\hat{z}_{j1}$ for all $j$
2: $z^+_{j1} = \hat{z}_{j1}$
3: for $k = 2, ..., n_{times}$ do
4:     Apply algorithm 5 (BE) to $x_{ijk}$ to obtain $\hat{z}_{jk}$
5:     $z^-_{jk} = \mathbb{M}(z^+_{j(k-1)})$ (Prediction)
6:     $z^+_{jk} = \hat{z}_{jk} + K \times (z^-_{jk} - \hat{z}_{jk})$ (Correction)
7:     Retrain $\mathbb{M}$
8: end for
```

Figure 4.2 shows the algorithm at a location j and at time t_k. The hidden variables include our objective variable z_{jk}, the observed variable is the data obtained from all the users at location j and time k, and the Bayesian estimation (BE) obtains an approximation to θ, H, and $\hat{z}_{jk}$. This last variable is used together with the prior prediction from the previous step ($z^{-}_{j(k-1)}$) in the Kalman estimation (KE) to obtain the posterior estimation (z^{+}_{jk}). The posterior is used together with the underlying model $\mathbb{M}$ to obtain a prior estimation for time t_{k+1}. The additional estimated variables from Bayesian Estimation ($\hat{h}_{*jk}$ and $\hat{\Theta}_{*}$) are used to improve the estimations of the Bayesian Estimation at the next time step. Algorithm 6 summarizes the whole process.

4.3 Experimental Results

In this section we describe the experiments performed. First, we describe a baseline method against which our methods were compared, and then present both a real-world case study and extensive simulations evaluating the impact of parameters and settings.

4.3.1 Compared Methods

Voting. For our baseline method, we consider the classic approach of *Voting*, where for each location and time, we simply count the number of reports and divide over all users. Since the number of users is in general much larger than the number of reports at a specific location and time, this ratio provides a very low number. We then need to set a threshold above which the method considers the label as true. In the simulation study, we optimized the threshold to have as many true positives as possible while for our real-world example, we set the threshold to 0, so that any positive report would make the method consider the event as true.

Truth Finder for Spatial Events (TSE). Truth Finder for Spatial Events [64] is a recent method that incorporates the probability of a user to be at a certain location, as well as the user's reliability. This method can effectively handle positive and missing reports to infer the truth through crowdsourced

Table 4.3: Precision, Recall, and F1 measure for the real-world data corresponding to figure 4.4(a)

		Voting			TSE			BE			BE+KE			BE+KE Pre		
Time		Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
Day 1	RH1	0.55	0.50	0.53	0.62	0.52	0.57	0.57	0.56	0.56	0.70	0.55	0.62	0.66	0.60	0.63
	RH2	0.43	0.33	0.38	0.49	0.36	0.41	0.46	0.37	0.41	0.62	0.56	0.59	0.67	0.64	0.66
Day 2	RH1	0.49	0.36	0.41	0.54	0.53	0.53	0.60	0.47	0.52	0.77	0.61	0.68	0.76	0.68	0.72
	RH2	0.39	0.36	0.37	0.47	0.41	0.44	0.52	0.37	0.43	0.67	0.67	0.67	0.77	0.72	0.74
Day 3	RH1	0.45	0.33	0.38	0.54	0.51	0.52	0.55	0.48	0.51	0.83	0.67	0.75	0.84	0.77	0.80
	RH2	0.48	0.34	0.40	0.60	0.43	0.50	0.50	0.47	0.49	0.79	0.72	0.75	0.82	0.82	0.82
Day 4	RH1	0.37	0.29	0.33	0.46	0.40	0.43	0.44	0.38	0.41	0.79	0.70	0.74	0.89	0.81	0.85
	RH2	0.34	0.31	0.33	0.44	0.38	0.41	0.40	0.39	0.40	0.80	0.72	0.75	0.89	0.81	0.85
Day 5	RH1	0.34	0.22	0.26	0.41	0.34	0.37	0.39	0.33	0.36	0.79	0.71	0.75	0.89	0.81	0.85
	RH2	0.32	0.24	0.28	0.49	0.36	0.42	0.42	0.38	0.40	0.83	0.70	0.76	0.89	0.81	0.85

data of spatial events. However, TSE does not consider a time dimension so it does not consider a time-dependent model for the changes in the events. To compare with our work, the whole TSE algorithm was run at each time-step with only the data available at that time step.

Bayesian Estimation (BE). In both the real-world case study and the simulation experiments, the Bayesian estimation method uses the graphical model presented in figure 4.1 and the algorithm described in Section 4.2.1. It does not assume knowledge of an event model and it is purely based on the reports from the users, but it takes into consideration the reports of previous time-steps.

Bayesian Estimation and Kalman Estimation (BE+KE).
This method is described in Section 4.2.1 and 4.2.2, and can train the event model $\mathbb{M}$ in real-time (BE+KE), or it can use existing historical data to train $\mathbb{M}$ (BE+KE pretrained). In the case of the simulation experiment, we also included a multidimensional model that considers the correlation between the locations as part of the event model $\mathbb{M}$ (BE+MD KE pretrained).

4.3.2 Real-World Case Study

Waze Reports. Waze is a navigation app where users can send reports from a predefined set of spatio-temporal events. The set of events includes potholes, accidents, and high traffic, and the reports are typically done while the users are driving, so in many occasions, users prefer not to send a report. For our real-world case study, we took a dataset of alert reports sent to the Waze application in the city of Boston, from 2/23/2015 to 3/1/2015. The dataset is publicly available[4] and includes date, time, latitude, longitude, anonymized user id, and type of report. This dataset provides a real-world application of spatio-temporal events with reports from users of varying and unknown reputation and undisclosed locations.

City Pothole data. For our ground truth, we took a dataset of closed pothole cases, which is publicly available in the city government website of Boston[5]. The dataset includes the date and time when the case was opened, the latitude, longitude, and the date and time when the case was closed. This dataset is updated every day since 2014, but we were only interested in those cases that intersected with the Waze reports. This includes cases that were opened before 2/23/2015 and closed on or after 3/1/2015, as well as those cases that were opened or closed during that time.

Figure 4.3(a) shows a time series of the potholes and the reports from the users in this period. The time series of the potholes was obtained from the Boston government website and the time series of the reports is from the Waze dataset. Each time slice represents a 4 hour period and we can see that the reports have peaks during the rush hours, and go down at night and on the weekends. The time series representing the potholes did not show such variation. The potholes that were not fixed during working hours remained open during the night.

[4]https://data.cityofboston.gov/Transportation/Waze-Alert-Data

[5]https://data.cityofboston.gov/city-services/closed-pothole-cases

Matching the reports with potholes on map. Taking the two potholes that were furthest away as corners of a square, we drew an $n \times n$ grid on the city of Boston. The number of grid cells was adjusted so that the cells were small enough to have most non-empty grid cells containing only one pothole, but big enough so that most non-empty grid cells had several reports in it. Figure 4.3(b) shows the distribution of potholes and reports on a 40 by 40 grid. The horizontal axis shows how many potholes and reports per individual cells there are, and the vertical axis is the count of such cells. The figure shows that there are several grid cells with only one pothole (bar plot on the left) and several grid cells with a large amount of reports (bar plot on the right). All of the non-empty grid cells were considered as locations and the rest were discarded from the analysis. Figure 4.3(c) shows a 10×10 close-up of the map of Boston with the officially reported potholes and the reports from the Waze users on a 4 hour period[6]. The 284 grid cells containing potholes or reports were considered as the locations ($n_{locs} = 284$) and the rest were discarded from the analysis. The dataset considered 2,396 different users (n_{users}) and 8,492 reports.

Compared Methods. We implemented the Bayesian and the Kalman Estimation model. For the event model in the Kalman Filter, we trained the method both in real time, and using previous data from the city government website of Boston. For the real-time training, we modeled the opening and closing of potholes as a time series and used an autoregressive moving average model (ARMA) to predict the next time-step [65]. This model has the advantage that it gives higher weight to the latest observations and less weight to older observations, and the number of observations to consider can be adjusted to the observations that are available at the time. On the other hand, the historical data provided by the city government showed that pot-

[6]Due to slight imprecision in the juxtaposition of images, there is a misalignment between the map and the points

holes are usually open for a few days, and that they are only fixed during working hours. With this data, we built a Markov model that showed that if a pothole was open, then with a very high probability it would remain open within the next 4 hours. If a pothole was closed, then with a very high probability it would remain closed. This gave a very accurate a priori prediction for each of the next time steps.

4.3.3 Evaluation metrics.

To evaluate the performance of the methods, we used *precision*, *recall*, and the *F1 measure*, which takes into account both precision and recall since $F1 = 2\frac{precision \times recall}{precision + recall}$. These metrics are commonly used to evaluate the performance of truth inference algorithms [54, 87, 23, 21].

4.3.4 Results

Table 4.3 shows the precision, recall, and F1 measure for the different methods at rush hours (RH). The table shows that precision tends to be higher, presumably because the missing data increases the number of false negatives, while false positives are less common. Figure 4.4(a) is a graph of the F1 measure, and we observe that the BE and TSE methods outperform in all cases the Voting method, but it is still very dependent on the data provided by the users. The seemingly low values for the performance, and the higher precision than recall is due to the missing values (e.g. fewer reports at night or low transited locations with few reports overall), which increases the number of false negatives, while false positives are less common. The Bayesian Estimation and Kalman Filter (BE+KE) method trained in real time has a performance very similar to the Bayesian Estimation in the first days, but once the time series has more data to perform the predictions and corrections, it performs better than Voting, TSE, and the BE method. If we use the historical data to train the model offline (BE+KE pre-trained) then the method starts similarly to the other three, but then it tends to give more weight to the a priori prediction over the observed and incomplete data.

Figure 4.4(b) shows a comparison of the methods throughout the process and it shows how the methods are dependent on the reports. In the case of Voting, there is a clear similarity with the reports of figure 4.3(a). Although the Bayesian Estimation and TSE outperform the Voting method, they are still very dependent on the available reports. The drops in the performance of Voting, BE, and TSE can be explained by the lack of reports during night time and the weekend. Both BE+KE and BE+KE pre-trained are more robust to this lack of reports due to the a priori estimate based on prediction.

Finally, a ROC curve in figure 4.4(c) shows a comparison of the different methods as we vary the threshold to determine true or false events. The area under the curve (AUC) is shown in parenthesis in the legend of the graph. We observe that the AUC of the BE+KE pre-trained method is 0.861, which makes this a very reliable method under these conditions. The graph shows how BE+KE retrained outperforms BE+KE, which in turn outperforms BE and TSE. Although BE is still dependent on the reports from the users, it outperforms the Voting method.

4.3.5 Simulation Experiment

To evaluate the impact on our methods caused by the different parameters, we also performed simulation experiments by generating synthetic data. We observed the effects in performance caused by changes in 1) number of users, 2) number of locations, 3) number of time slices, and 4) reliability traits of users.

Simulation Setup

We tried different parameters for the n_{users} and n_{locs}. We fixed $n_{times} = 24$ to represent a scenario where each time period represents an hour of the day. For each user u_i, we generate their reliability traits $\theta_i = (\alpha_i, \beta_i, \gamma_i)$ where each of these parameters is a value between 0 and 1. We simulate the reliability traits of the users with a Beta distribution (the conjugate prior of the Bernoulli distribution) as was done in [64]. Then we simulated the popularity of the locations (g_j with $j = 1, ..., n_{locs}$) by drawing from a power law distribution [64] and generated variable H using this data and the definition of popularity of a location (section 4.2.1). Any user had a g_j chance of visiting location l_j at some point in time, and once a user was at a certain location, then the same user could not be at any other location at the same time.

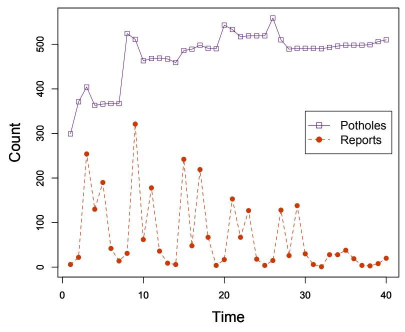
To model the true label of the events, we used two models. Under the first model, we assume that the locations are independent of each other and we use a different Markov model for each location, where the states of the model are *True* or *False* and we simulate with different transition probabilities. In general, these probabilities can be learned in real time **(BE+KE)** as well as using historical data **(BE+KE pre-trained)**. The second model corresponded to the case in which the events at the locations are correlated. Such a scenario could be for example modelling traffic, where high traffic on one location could be correlated with high traffic on a nearby location. The correlation matrix is learned offline and the true label of the events is simulated accordingly. We refer to this multidimensional method as **BE+MD KE pretrained**. Finally, to compare the effect that the Kalman estimation method has on our BE method, we replaced the BE with another method for truth inference (the previously mentioned TSE) together with the Kalman estimation and used it with the simulated data **(TSE+KE)**

Once we have simulated H and Z, we use Θ_i to simulate the reports from the users, according to equations 4.1, 4.2, and 4.3. For each user u_i at location l_j and time t_k, the probability that $x_{ijk} = 1$ is drawn from a Bernoulli distribution with parameter α_i, β_i, or γ_i, depending on the values of h_{ijk} and z_{jk}. Algorithm 7 summarizes the simulation procedure.

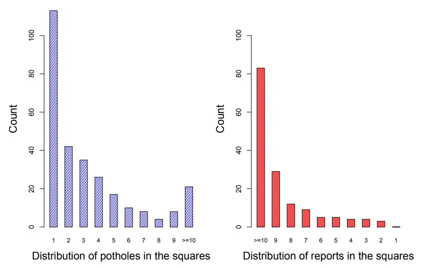
Impact of user reliability

Figure 4.5 shows the results of the simulations as we varied different values of Θ. We tried different values for α, β, and γ, which test different degrees of reliability (passiveness, trustworthiness, maliciousness), and the figures in 4.5 are representative of all the different scenarios. In figure 4.5(a), we fixed the value of β to 0.25 and γ to 0.05 (i.e. low values) and observed the performance of the different values as a function of α. A low value of both β and α represents the situation in which the users are not very participative, and they do not often report the event regardless of it being true or false. In this case, we have several missing values. As α increases with a fixed and low β, the users become more trustworthy, and only report events when the event is true, increasing the number of true positive values without increasing the false positives or false negatives, and therefore increasing the F1 score.

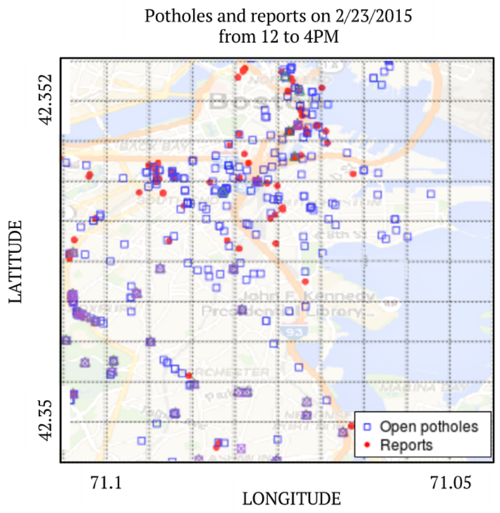
In figure 4.5(b), we fixed the values for α to 0.85, left γ as before, and observe the performance of the methods as we vary β. In this scenario, there is a slight decrease in the performance of the methods as β increases. A high value of α and β corresponds to the scenario of aggressive users who report events as true even if they are ambiguous or flat out false. There is an increase of true positives, but also of false positives, which makes the value of $F1$ to decrease slightly. For figure 4.5(c), we kept the values of α at 0.85 and β at 0.25, and modified the values of γ. For low values of γ, we observe a scenario with trustworthy users, but as γ increases, so do their reports when the users are not at the specific location and time. This corresponds to the scenario of malicious users, which increases the values of the false positives and false negatives, decreasing the overall performance of the methods.



(a) Potholes and user reports from 2/23/2015 to 3/1/2015. Each time slice is 4 hours long

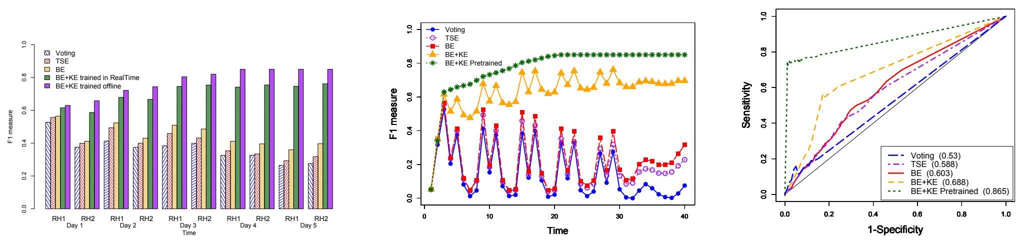


(b) Using a 40x40 grid, most non-empty grid cells have only 1 pothole (blue bar on the left-hand side of the graph) while several grid cells have a considerable amount of reports (red bar on the right)



(c) Map of Boston with officially reported potholes and reports by Waze users. The new and closed potholes relate only to this time period, while the open potholes were reported before and closed after this 4 hour period.

Figure 4.3: Analysis of the datasets used in the real-world application



(a) Comparison of the different methods at rush hours along 5 different days, where RH1 is 8am to 12pm and RH2 is 4pm to 8pm

(b) Different methods compared throughout the whole timeline

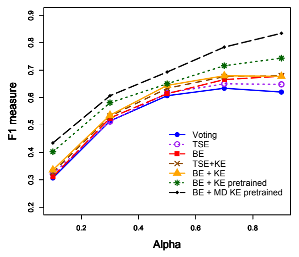
(c) ROC curve comparing the different methods. The AUC is shown in parenthesis for each method

Figure 4.4: Comparison of the methods using real-world data of potholes and Waze reports

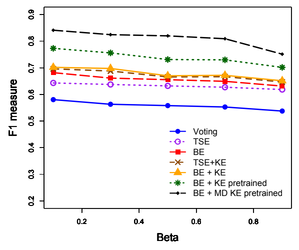
In general, we can see that BE and BE+MD KE outperforms the other methods in all scenarios because it models the spatio-temporal correlations of the events explicitly. TSE+KE also outperforms TSE, which highlights the benefit of using the event model and the Kalman Estimation.

Impact of number of users, locations and time periods.

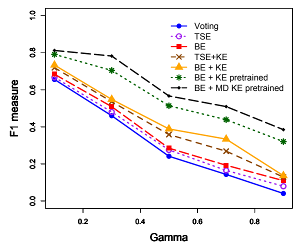
Next, we tried our methods using different number of users, locations, and time periods. We tried different values for each, and Figure 4.6(a) shows the performance of the methods as a time series with $n_{times} = 24$. For the first few time periods, we observe that the BE and TSE method outperform the rest. However, as we get more reports and are able to train the Kalman Estimation, the four methods that use the Kalman Estimation started performing better. In particular, we see that TSE+KE performs better in time (and better than TSE) since the Kalman estimation corrects the estimated values. TSE is not dependent on time and performs equally good at each time-step, while BE performs better in time since it uses previous reports to improve the estimation parameters at each time-step.



(a) Performance when varying α and fixing $\beta = 0.25$, $\gamma = 0.05$.

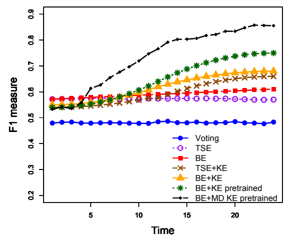


(b) Performance when varying β and fixing $\alpha = 0.85$, $\gamma = 0.05$.

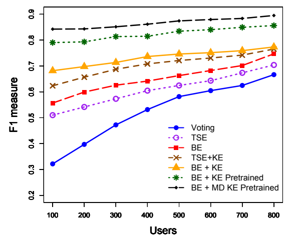


(c) Performance when varying γ and fixing $\alpha = 0.85$, $\beta = 0.25$.

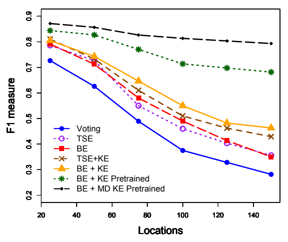
Figure 4.5: Results of simulations as a function of the reliability model for the users



(a) Evaluation of the performance of the methods as a function of time



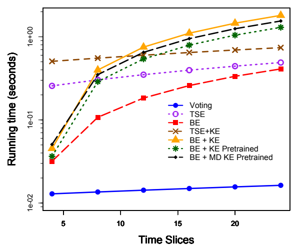
(b) Number of users versus F1



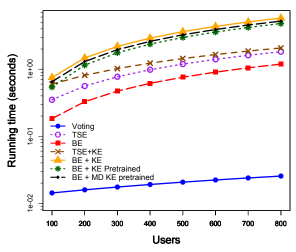
(c) Number of locations versus F1

Figure 4.6: Results of simulations

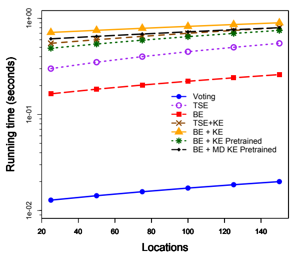
In figures 4.6(b) and 4.6(c), we again fixed the value of n_{times} to 24, and observed the performance of the methods as we varied the number of users and locations, respectively. Voting, BE, and TSE are the most sensitive to the reported data, while the methods that use the Kalman Estimation can depend more on the Prediction and Correction phases and hence outperform the classic and BE methods. Figure 4.6(b) shows that the more users we include, the methods will show an increase in performance. Figure 4.6 shows that if we keep all the other variables constant, and increase the number of



(a) Evaluation of runtime as a function on the number of time slices



(b) Evaluation of runtime as a function of the number of users



(c) Number of runtime as a function of the number of locations

Figure 4.7: Runtime of the simulations

locations, the performance will decrease.

Runtime analysis

Figure 4.7 shows logarithmic plots of the runtime of the methods as a function of number of time slices, users, and locations. The experiments were run in a machine with Intel Core i7 CPU at 4.00GHz and 64 Gb of memory. In figure 4.7(a), we observe that Voting and TSE are the least affected by adding more time slices, since each time slice is evaluated independently and the increase in runtime is due to the increase of evaluation slices. BE performs faster than TSE since it considers all the time slices at the same time, but it is affected more by the increase of time slices since it requires convergence of more values. Similarly, the methods based on the Kalman estimation require evaluation of more time slices and convergence of the BE model at each time step. Figure 4.7(b) shows how the methods are affected by increasing the number of users and in general we see an increase in the runtime on all the methods. A similar situation occurs when increasing the number of locations (figure 4.7(c)). In all cases we observe that it is better to have a pre-trained model to use with the Kalman filter.

Algorithm 7 Algorithm to generate the simulation variables

```
1:  Determine the values of $n_{users}$, $n_{locs}$, and $n_{times}$
2:  For each user $u_i$, determine $\theta_i$ either by fixing the values or with a Beta distribution.
3:  For each location $l_j$, determine its popularity $g_j$ from a power-law distribution
4:  Determine a transition model $\mathbb{M}$ and an initial probability for the true label of the events. The model can be unidimensional or multidimensional if there is a correlation between the events.
5:  **for** $j = 1, ..., n_{locs}$ **do**
6:      Set $z_{j1}$ using the initial probabilities
7:      **for** $k = 1, ..., n_{times}$ **do**
8:          Use the location popularity to determine $H$
9:          $z_{j(k+1)} = \mathbb{M}(z_{jk}, ..., z_{j1})$
10:         **if** $z_{jk} = 1$ and $h_{ijk} = 1$ **then**
11:             $X_{ijk} \sim \text{Ber}(\alpha_i)$
12:         **end if**
13:         **if** $z_{jk} = 0$ and $h_{ijk} = 1$ **then**
14:             $X_{ijk} \sim \text{Ber}(\beta_i)$
15:         **end if**
16:         **if** $h_{ijk} = 0$ **then**
17:             $X_{ijk} \sim \text{Ber}(\gamma_i)$
18:         **end if**
19:     **end for**
20: **end for**
```

Chapter 5

Conclusion

In this chapter, we summarize our results and contributions, and outline areas of possible future research.

5.1 Summary

In this work, we have proposed a task distribution model in spatial crowdsourcing that consists in a recommender system that balances the user and the task point of view. Mapping each user to a set of tasks so that the expected utility for the users, and the sum of probabilities that all the tasks are performed is maximized, is an NP-hard problem. We have proposed several algorithms to approximate the exact solution, which is intractable, and have tested our solutions with both simulated data and real-world data. In the latter case, we also predicted the location and benefit of the tasks using non-negative tensor factorization.

We have also proposed a solution to the truth-inference problem that determines the true label of an event from missing or inconsistent data. The solution uses recursive Bayesian estimation and it is later enhanced by using the Kalman filter and the event model. We ran our method both in simulated data and real-world data and presented the results.

5.2 Recommender Systems in Spatial Crowdsourcing

We have proposed a general framework for recommendations in spatial crowdsourcing by defining the system utility as a balance between the expected utility for the users and the sum of probabilities of completion for the tasks. These utilities are usually in contention since the more users are recommended a same task, the higher the probability of the task being completed, but the lower expected utility for the users. We proved that this problem is NP-hard combinatorial optimization problem and that obtaining the exact solution can be intractable for real-world applications. We suggested several algorithms that obtain an approximated solution in less time, and observe the trade-off between utility and running time.

The greedy algorithm had the lowest running time, and obtains a relatively good user utility. However, it does not consider the interactions between users, and can suggest the same task to several users. This increases the probability that that task is performed, but decreases the probability that several other tasks would be performed. This is also reflected in the user utility, since each user has a lower probability of performing that task. If the application is mostly interested in the user utility (i.e. $\alpha = 1$) and speed, then this could be a viable solution.

On the other hand, the ordered-progressive method, which is a slight modification of the progressive method, had both the highest user and task utility (and therefore system utility). The progressive method optimizes one user at a time, and once a user is optimized, it does not go back and change any recommendation, even though it might yield a higher utility. Ordered-progressive first orders the users according to their individual utility, and as we proved in Chapter 3, the utility yielded by this method is not lower than the utility from the progressive method. Ordered-progressive, however, cannot guarantee optimality, but can be used if the application is interested in obtaining a high user and task utility.

Hybrid methods are well-suited for larger problems, where the running time for ordered-progressive is still unacceptable. These methods group the users and the tasks into smaller sets and then perform (ordered) progressive method. The utility is slightly lower than with ordered-progressive, since not all users and tasks are considered at the same time, but their running time make them a viable solution for real-world problems.

Non-negative tensor factorization proved to be a viable tool for prediction and analysis. In terms of analysis, plotting the first few components of each mode allows us to learn more about the data. For this work, we used two modes for location (grid x and grid y). We could have also used only one mode for the locations, described for example with a Hilbert curve to capture as most as possible the interactions of nearby locations. However, using two modes for location not only allows us to capture the interactions between the nearby locations, but it is also useful in terms of analysis to determine the locations with the highest demand and benefit for performing the tasks.

For future work, this general framework can be extended to include different aspects from both the users' and the tasks' point of view. For example, from the users' point of view, the users could choose between different types of tasks according to their preference or their skills. This would require some changes in the individual utility function. From the tasks' point of view, we could include a penalty for not recommending a task to also maximize the number of tasks that are recommended and not just the probability of being chosen.

5.3 Truth Inference in Spatial Crowdsourcing

We have proposed a probabilistic graphical model for truth-inference in a spatio-temporal scenario that incorporates the user's reliability model and does not need to constantly track their location. The method also combines space-state model based prediction of a next time period and fusion with currently available reports. One of the advantages of this approach is that the estimations can be done in real time. If an observation has not arrived yet, we can use the a priori prediction as our estimation, making it suitable for streaming data, and once the information arrives we can update to get a better estimation. The method is versatile to the available data and can easily incorporate different prediction models. Experimental results on both real world data and simulated data show F1 scores that increase as more time periods are accessible and better prediction models are used, making it less dependable on the observations like the classic methods. The algorithm with a pre-trained model from historical data has a better performance, and it is preferable to use when the data is available.

The applicability of the model can be extended by considering the following: **Events could have more than two labels.** For example, users could report if there is medium or high traffic. In such a case, users would need to be able to report on each of the different states and the model would need to be extended to include the probability for each state. If n_s is the number of states, then table 4.2 requires the computation of $n_s \times (n_s - 1) + n_s - 1 = n_s^2 - 1$, since when $h_{ijk} = 1$, for each of the n_s states we require $n_s - 1$ probabilities (since the last one can be inferred from the rest) and for $h_{ijk} = 0$ we require $n_s - 1$ probabilities. The reliability model for the users would also require more parameters to capture how a user would react under different circumstances. Depending on the application and the assumptions, the number of parameters can go from $2n_s - 1$ (assuming e.g.

$P(x_{ijk} = s_1|z_{jk} = s_3) = P(x_{ijk} = s_2|z_{jk} = s_3)$ with s_1, s_2, s_3 non-default states) to $n_s^2 - 1$ (if no such assumptions are made).

Continuous variables could be discretized into ranges. For example, the users could report that price of gas is either below \$1.50, between \$1.50 and \$2.50, or higher than \$2.50. By having a default state (e.g. higher than \$2.50), this scenario would be reduced to the previous one.

Users could report a "negative" state. For example, users could report that there is no accident at a certain location. This scenario eliminates the default state and changes the interpretability of the missing reports, since they would only be the result of lack of participation or because the user was not at the specified time and location. A negative report could be either because an event was incorrectly reported or because it is no longer true.

5.4 Future Work

The current state of spatial crowdsourcing has still several areas of opportunities that should be addressed in the near future. In terms of user-task mapping, several task assignment problems have been studied where different restrictions are involved (e.g. budget constraints, unknown user reliability or skills, required minimum accuracy). The problem is more complicated when crowdsourcing is spatial, since there could be other additional restrictions such as expiration time for the tasks, nearby users are not qualified, tasks are also moving, or the users only have certain available time.

In terms of task prediction, tensor factorization offers dimensionality reduction, scalability, and in the case of non-negative tensor factorization, also interpretability of the data. However, there are still several open questions about tensor analysis and factorization. Although tensors are a generalization of matrices, not all of the algorithms and well-established concepts in matrix analysis are easily extendible to tensors. For example, there is still not an optimal way to determine the rank of a tensor, nor a way to determine what the best rank-n approximation to a tensor [49]. Tensor analysis is an area of increasing interest due to its wide variety of possible applications and the number of open research questions.

Truth inference algorithms have been widely studied, but mostly for non-spatial tasks. Zheng et. al. [87] did a comparison of 17 different non-spatial truth inference algorithms and determined that no algorithm dominates over all others, but give suggestions given the type of data and the amount of available answers.

A recent line of research in truth inference refers to attacks from malicious entities, who might want to maximize the number of changes from one label to another while at the same time maximizing the malicious users' reliability. This could be done through a "poison attack", where the attacker, disguised as a regular user, injects data in such a way that its reliability is increased by correctly performing uncontroversial tasks, and maliciously sends an incorrect label on contentious tasks. Another possibility for an attack is to simulate several users that provide a wrong or random answer, making the truth inference algorithm return a noisy response. An open area of research consists in building robust truth inference algorithms that can be immune to these type of attacks.

Spatial tasks have a higher complexity because they can be changing location, their value, or have an expiration time. Furthermore, users are also moving, which makes the task assignment problem much more complex. One of the biggest challenges in truth inference in spatial crowdsourcing is designing the system in such a way that the privacy of the users is guaranteed. Due to the nature of spatial crowdsourcing, location attacks can occur, and an attacker might discover the user's home or work address or some patterns in its lifestyle. Privacy preserving techniques such as spatial cloaking [48] require that the user's location is general enough to protect its privacy, but specific enough to assign the right task. Since the functionality of truth inference algorithms depend on the participation of users, its performance will decrease if the privacy of the users cannot be guaranteed.

Appendix

6.1 Tighter Bounds for User and Task Utilities

Users' utility: The user's utility can be obtained from equation 3.3 by setting $\alpha = 1$. In that case, we obtain $f_u(R) = \sum_{i=1}^{n_u} \sum_{s_j \in R_i} \lambda_i P(u_i \to s_j) Y_{ij}$. To set a bound for $f_u(R)$, notice that $0 \leq \sum_{s_j \in R_i} P(u_i \to s_j) \leq 1$ for all $u_i \in U$. Let $Y_i^+ = \max_j Y_{ij}$ for each $u_i \in U$. Then $0 \leq \sum_{s_j \in R_i} \lambda_i P(u_i \to s_j) Y_{ij} \leq \lambda_i \sum_{s_j \in R_i} P(u_i \to s_j) Y_i^+ \leq \lambda_i Y_i^+ \sum_{s_j \in R_i} P(u_i \to s_j) \leq \lambda_i Y_i^+$. Therefore, $0 \leq f_u(R) = \sum_{i=1}^{n_u} \sum_{s_j \in R_i} \lambda_i P(u_i \to s_j) Y_{ij} \leq \sum_{i=1}^{n_u} \lambda_i Y_i^+$. This bound could only be reached if the cost of opportunity is 0 (i.e. $\sum_{s_j \in R_i} P(u_i \to s_j) = 1$) for all users, which is never the case, unless all users assign no value to the distance needed to travel to the task ($\omega_i = 0$) and the benefit for performing the next best task that was not recommended is 0 (i.e. $B(u_i, s_{(n_r+1)}) = 0$ for all u_i). Only in this extreme case would we have that $Y_{i(n_r+1)} = 0$ and we could reach the upper bound. Define $B_u := \sum_{i=1}^{n_u} \lambda_i Y_i^+$, so that $0 \leq f_u(R) \leq B_u$.

Tasks' utility: Similarly to the definition of the users' utility, setting $\alpha = 0$ in equation 3.3 yields $f_s(R) := \sum_{s_j \in S_R} \gamma_j P(s_j)$, where $S_R = \{s_j \in R | s_j \in R_i \text{ for some } R_i \in R\} \subseteq S$. Since $0 \leq P(s_j) \leq 1$ for all $s_j \in S$, then $0 \leq \gamma_j P(s_j) \leq \gamma_j$ and $0 \leq f_s(R) = \sum_{s_j \in S_R} \gamma_j P(s_j) \leq \sum_{sj \in S_R} \gamma_j$. Since the tasks' priorities are normalized, $\sum_{sj \in S_R} \gamma_j = 1$ iff $S_R = S$ (i.e. all tasks were recommended). Similarly, the upper bound could only be achieved if all the tasks had a probability one of being chosen. This again requires the cost of opportunity to be 0, and that we only have one recommendation per user (i.e. an assignment and not a recommendation). Define $B_s = \sum_{sj \in S_R} \gamma_j$, so that $0 \leq f_s(R) \leq B_s$.

Bibliography

[1] Charu C Aggarwal and Tarek Abdelzaher. Social sensing. In *Managing and Mining Sensor Data*, pages 237–297. Springer, 2013.

[2] Vamshi Ambati, Stephan Vogel, and Jaime G Carbonell. Towards task recommendation in micro-task markets. *Human computation*, 11:11, 2011.

[3] Daniel Ashbrook and Thad Starner. Using gps to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous computing*, 7(5):275–286, 2003.

[4] Laure Berti-Equille, Anish Das Sarma, Xin Dong, Amélie Marian, and Divesh Srivastava. Sailing the information ocean with awareness of currents: Discovery and application of source dependence. In *CIDR*. Citeseer, 2009.

[5] Lindsay Brown, Bernard Grundlehner, Jef van de Molengraft, Julien Penders, and Bert Gyselinckx. Body area network for monitoring autonomic nervous system responses. In *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on*, pages 1–3. IEEE, 2009.

[6] Nirupama Bulusu, Chun Tung Chou, Salil Kanhere, Yifei Dong, Shitiz Sehgal, David Sullivan, and Lupco Blazeski. Participatory sens-

ing in commerce: Using mobile camera phones to track market price dispersion. In *Proceedings of the International Workshop on Urban, Community, and Social Applications of Networked Sensing Systems (UrbanSense 2008)*, pages 6–10, 2008.

[7] Xin Cao, Gao Cong, and Christian S Jensen. Mining significant semantic locations from gps data. *Proceedings of the VLDB Endowment*, 3(1-2):1009–1020, 2010.

[8] CHEN CEN, Shih-Fen Cheng, Hoong Chuin Lau, and Archan Misra. Towards city-scale mobile crowdsourcing: Task recommendations under trajectory uncertainties. 2015.

[9] Kai-Wei Chang, Scott Wen-tau Yih, Bishan Yang, and Chris Meek. Typed tensor decomposition of knowledge bases for relation extraction. 2014.

[10] Zhao Chen, Rui Fu, Ziyuan Zhao, Zheng Liu, Leihao Xia, Lei Chen, Peng Cheng, Caleb Chen Cao, Yongxin Tong, and Chen Jason Zhang. gmission: A general spatial crowdsourcing platform. *Proceedings of the VLDB Endowment*, 7(13):1629–1632, 2014.

[11] Peng Cheng, Xiang Lian, Zhao Chen, Rui Fu, Lei Chen, Jinsong Han, and Jizhong Zhao. Reliable diversity-based spatial crowdsourcing by moving workers. *Proceedings of the VLDB Endowment*, 8(10):1022–1033, 2015.

[12] Eric C Chi and Tamara G Kolda. On tensors, sparsity, and nonnegative factorizations. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1272–1299, 2012.

[13] D. Christin, A. Reinhardt, S. S. Kanhere, and M. Hollick. A survey on privacy in mobile participatory sensing applications. *Journal of Systems and Software*, 84(11):1928–1946, 2011.

[14] Noel Cressie and Christopher K Wikle. Space-time kalman filter. *Encyclopedia of environmetrics*, 2002.

[15] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma. Prism: Platform for remote sensing using smartphones. pages 63–76. in Proceedings of the 8th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys), 2010.

[16] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.

[17] Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the 21st international conference on World Wide Web*, pages 469–478. ACM, 2012.

[18] Dingxiong Deng, Cyrus Shahabi, and Ugur Demiryurek. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *Proceedings of the 21st acm sigspatial international conference on advances in geographic information systems*, pages 324–333. ACM, 2013.

[19] Linda Deng and Landon P Cox. Livecompare: grocery bargain hunting through participatory sensing. In *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, page 4. ACM, 2009.

[20] Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux. Pick-a-crowd: tell me what you like, and i'll tell you what to

do. In *Proceedings of the 22nd international conference on World Wide Web*, pages 367–374. ACM, 2013.

[21] Xin Luna Dong, Laure Berti-Equille, Yifan Hu, and Divesh Srivastava. Global detection of complex copying relationships between sources. *Proceedings of the VLDB Endowment*, 3(1-2):1358–1369, 2010.

[22] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. Integrating conflicting data: the role of source dependence. *Proceedings of the VLDB Endowment*, 2(1):550–561, 2009.

[23] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. Truth discovery and copying detection in a dynamic world. *Proceedings of the VLDB Endowment*, 2(1):562–573, 2009.

[24] Daniel M Dunlavy, Tamara G Kolda, and Evrim Acar. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(2):10, 2011.

[25] Shane B Eisenman, Emiliano Miluzzo, Nicholas D Lane, Ronald A Peterson, Gahng-Seop Ahn, and Andrew T Campbell. Bikenet: A mobile sensing system for cyclist experience mapping. *ACM Transactions on Sensor Networks (TOSN)*, 6(1):6, 2009.

[26] Liyue Fan, Luca Bonomi, Li Xiong, and Vaidy Sunderam. Monitoring web browsing behavior with differential privacy. In *Proceedings of the 23rd international conference on World wide web*, pages 177–188. ACM, 2014.

[27] Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. Annotating named entities in twitter data with crowdsourcing. In *Proceedings of the NAACL HLT 2010 Workshop*

on Creating Speech and Language Data with Amazon's Mechanical Turk, pages 80–88. Association for Computational Linguistics, 2010.

[28] André Sales Fonteles, Sylvain Bouveret, and Jérôme Gensel. Towards matching improvement between spatio-temporal tasks and workers in mobile crowdsourcing market systems. In *Proceedings of the third acm sigspatial international workshop on mobile geographic information systems*, pages 43–50. ACM, 2014.

[29] André Sales Fonteles, Sylvain Bouveret, and Jérôme Gensel. Heuristics for task recommendation in spatiotemporal crowdsourcing systems. In *Proceedings of the 13th International Conference on Advances in Mobile Computing and Multimedia*, pages 1–5. ACM, 2015.

[30] Alban Galland, Serge Abiteboul, Amélie Marian, and Pierre Senellart. Corroborating information from disagreeing views. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 131–140. ACM, 2010.

[31] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. De-anonymization attack on geolocated data. *Journal of Computer and System Sciences*, 2014.

[32] Raghu K Ganti, Fan Ye, and Hui Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11), 2011.

[33] Daniel A Garcia-Ulloa, Li Xiong, and Vaidy Sunderam. Truth discovery for spatio-temporal events from crowdsourced data. *Proceedings of the VLDB Endowment*, 10(11):1562–1573, 2017.

[34] David Geiger. *Personalized Task Recommendation in Crowdsourcing Systems*. Springer, 2016.

[35] Yanmin Gong, Yuanxiong Guo, and Yuguang Fang. A privacy-preserving task recommendation framework for mobile crowdsourcing. In *Global Communications Conference (GLOBECOM), 2014 IEEE*, pages 588–593. IEEE, 2014.

[36] Yanmin Gong, Lingbo Wei, Yuanxiong Guo, Chi Zhang, and Yuguang Fang. Optimal task recommendation for mobile crowdsourcing with privacy control. *IEEE Internet of Things Journal*, 3(5):745–756, 2016.

[37] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42. ACM, 2003.

[38] Ramaswamy Hariharan and Kentaro Toyama. Project lachesis: parsing and modeling location histories. In *International Conference on Geographic Information Science*, pages 106–124. Springer, 2004.

[39] Joyce C Ho, Joydeep Ghosh, Steve R Steinhubl, Walter F Stewart, Joshua C Denny, Bradley A Malin, and Jimeng Sun. Limestone: High-throughput candidate phenotype generation via tensor factorization. *Journal of biomedical informatics*, 52:199–211, 2014.

[40] Baik Hoh, Tingxin Yan, Deepak Ganesan, Kenneth Tracton, Toch Iwuchukwu, and Juong-Sik Lee. Trucentive: A game-theoretic incentive platform for trustworthy mobile crowdsourcing parking services. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pages 160–166. IEEE, 2012.

[41] Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan, and Samuel Madden. Cartel: a distributed mobile sensor computing system. In

Proceedings of the 4th international conference on Embedded networked sensor systems, pages 125–138. ACM, 2006.

[42] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[43] Panos Kalnis, Gabriel Ghinita, Kyriakos Mouratidis, and Dimitris Papadias. Preventing location-based identity inference in anonymous spatial queries. *Knowledge and Data Engineering, IEEE Transactions on*, 19(12):1719–1733, 2007.

[44] A. Kansal, M. Goraczko, and F. Zhao. Building a sensor network of mobile phones. pages 547–548. in Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN), 2007.

[45] David R Karger, Sewoong Oh, and Devavrat Shah. Budget-optimal crowdsourcing using low-rank matrix approximations. In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, pages 284–291. IEEE, 2011.

[46] David R Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. In *Advances in neural information processing systems*, pages 1953–1961, 2011.

[47] Leyla Kazemi and Cyrus Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *Proceedings of the 20th international conference on advances in geographic information systems*, pages 189–198. ACM, 2012.

[48] Ali Khoshgozaran and Cyrus Shahabi. Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In *Advances in Spatial and Temporal Databases*, pages 239–257. Springer, 2007.

[49] Tamara G Kolda. Orthogonal tensor decompositions. *SIAM Journal on Matrix Analysis and Applications*, 23(1):243–255, 2001.

[50] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[51] John Krumm. Inference attacks on location tracks. In *Pervasive Computing*, pages 127–143. Springer, 2007.

[52] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[53] Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1187–1198. ACM, 2014.

[54] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han. A Survey on Truth Discovery. *ArXiv e-prints*, May 2015.

[55] Yaliang Li, Qi Li, Jing Gao, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. On the discovery of evolving truth. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 675–684. ACM, 2015.

[56] Yu Li, Man Lung Yiu, and Wenjian Xu. Oriented online route recommendation for spatial crowdsourcing task workers. In *International Symposium on Spatial and Temporal Databases*, pages 137–156. Springer, 2015.

[57] Lin Liao, Donald J Patterson, Dieter Fox, and Henry Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171(5-6):311–331, 2007.

[58] Christopher H Lin, Ece Kamar, and Eric Horvitz. Signals in the silence: Models of implicit feedback in a recommendation system for crowdsourcing. In *AAAI*, pages 908–915, 2014.

[59] Hong Lu, Nicholas D Lane, Shane B Eisenman, and Andrew T Campbell. Bubble-sensing: Binding sensing tasks to the physical world. *Pervasive and Mobile Computing*, 6(1):58–71, 2010.

[60] Prashanth Mohan, Venkata N Padmanabhan, and Ramachandran Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 323–336. ACM, 2008.

[61] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816, 2011.

[62] Shuxin Nie, Abhimanyu Das, Evgeniy Gabrilovich, Wei-Lwun Lu, Boris Mazniker, and Chris Schilling. Steps: Predicting place attributes via spatio-temporal analysis. *arXiv preprint arXiv:1610.07090*, 2016.

[63] Robin Wentao Ouyang, Animesh Srivastava, Prithvi Prabahar, Romit Roy Choudhury, Merideth Addicott, and F Joseph McClernon. If you see something, swipe towards it: crowdsourced event localization using smartphones. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 23–32. ACM, 2013.

[64] Robin Wentao Ouyang, Mani Srivastava, Alice Toniolo, and Timothy J Norman. Truth discovery in crowdsourced detection of spatial events. In

Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pages 461–470. ACM, 2014.

[65] Bei Pan, Ugur Demiryurek, and Cyrus Shahabi. Utilizing real-world transportation data for accurate traffic prediction. In *2012 IEEE 12th International Conference on Data Mining*, pages 595–604. IEEE, 2012.

[66] Bei Pan, Yu Zheng, David Wilkie, and Cyrus Shahabi. Crowd sensing of traffic anomalies based on human mobility and social media. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 344–353, 2013.

[67] Layla Pournajaf, Daniel A Garcia-Ulloa, Li Xiong, and Vaidy Sunderam. Participant privacy in mobile crowd sensing task management: a survey of methods and challenges. *ACM SIGMOD Record*, 44(4):23–34, 2016.

[68] Layla Pournajaf, Li Xiong, Vaidy Sunderam, and Slawomir Goryczka. Spatial task assignment for crowd sensing with cloaked locations. In *Proceedings of the 2014 International Conference on Mobile Data Management. IEEE*, 2014.

[69] Chenxi Qiu, Anna C Squicciarini, Barbara Carminati, James Caverlee, and Dev Rishi Khare. Crowdselect: Increasing accuracy of crowdsourcing tasks through behavior prediction and user selection. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 539–548. ACM, 2016.

[70] Steffen Schnitzer, Svenja Neitzel, Sebastian Schmidt, and Christoph Rensing. Perceived task similarities for task recommendation in crowdsourcing systems. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 585–590. International World Wide Web Conferences Steering Committee, 2016.

[71] Steffen Schnitzer, Christoph Rensing, Sebastian Schmidt, Kathrin Borchert, Matthias Hirth, and Phuoc Tran-Gia. Demands on task recommendation in crowdsourcing platforms-the workers perspective. In *ACM RecSys 2015 CrowdRec Workshop, Vienna*, 2015.

[72] M. Shin, C. Cornelius, D. Peebles, A. Kapadia, D. Kotz, and N. Triandopoulos. Anonysense: A system for anonymous opportunistic sensing. *Journal of Pervasive and Mobile Computing*, 7(1):16–30, 2010.

[73] Ajit P Singh and Geoffrey J Gordon. A unified view of matrix factorization models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 358–373. Springer, 2008.

[74] Hien To, Liyue Fan, Luan Tran, and Cyrus Shahabi. Real-time task assignment in hyperlocal spatial crowdsourcing under budget constraints. In *Pervasive Computing and Communications (PerCom), 2016 IEEE International Conference on*, pages 1–8. IEEE, 2016.

[75] Hien To, Gabriel Ghinita, Liyue Fan, and Cyrus Shahabi. Differentially private location protection for worker datasets in spatial crowdsourcing. *IEEE Transactions on Mobile Computing*, 16(4):934–949, 2017.

[76] Güliz Seray Tuncay, Giacomo Benincasa, and Ahmed Helmy. Autonomous and distributed recruitment and data collection framework for opportunistic sensing. *ACM SIGMOBILE Mobile Computing and Communications Review*, 16(4):50–53, 2013.

[77] Dong Wang, Tarek Abdelzaher, Lance Kaplan, and Charu C Aggarwal. Recursive fact-finding: A streaming approach to truth estimation in crowdsourcing applications. In *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*, pages 530–539. IEEE, 2013.

[78] Dong Wang, Lance Kaplan, Tarek Abdelzaher, and Charu C Aggarwal. On credibility estimation tradeoffs in assured social sensing. *Selected Areas in Communications, IEEE Journal on*, 31(6):1026–1037, 2013.

[79] Dong Wang, Lance Kaplan, Hieu Le, and Tarek Abdelzaher. On truth discovery in social sensing: A maximum likelihood estimation approach. In *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, pages 233–244. ACM, 2012.

[80] Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. Crowder: Crowdsourcing entity resolution. *Proceedings of the VLDB Endowment*, 5(11):1483–1494, 2012.

[81] Yu-Xiong Wang and Yu-Jin Zhang. Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1336–1353, 2013.

[82] Yang Ye, Yu Zheng, Yukun Chen, Jianhua Feng, and Xing Xie. Mining individual life pattern based on location history. In *Mobile Data Management: Systems, Services and Middleware, 2009. Tenth International Conference on*, pages 1–10. IEEE, 2009.

[83] Xiaoxin Yin, Jiawei Han, and Philip S Yu. Truth discovery with multiple conflicting information providers on the web. *Knowledge and Data Engineering, IEEE Transactions on*, 20(6):796–808, 2008.

[84] Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. Task recommendation in crowdsourcing systems. In *Proceedings of the first international workshop on crowdsourcing and data mining*, pages 22–26. ACM, 2012.

[85] Bo Zhao, Benjamin IP Rubinstein, Jim Gemmell, and Jiawei Han. A bayesian approach to discovering truth from conflicting sources for data integration. *Proceedings of the VLDB Endowment*, 5(6):550–561, 2012.

[86] Zhou Zhao, James Cheng, and Wilfred Ng. Truth discovery in data streams: A single-pass probabilistic approach. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1589–1598. ACM, 2014.

[87] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. Truth inference in crowdsourcing: Is the problem solved? *Proceedings of the VLDB Endowment*, 10(5), 2017.