TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technology

Bilal Masud 195651IASM

Serendipity-based Point of Interest Recommendation System

Master's Thesis

Supervisor
Sadok Ben Yahia
Professor
Co-Supervisor
Imen Ben Sassi
Post-Doc Researcher

Tallinn 2021

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Bilal Masud 195651IASM

Vedamispõhine Huvipunktide Soovitussüsteem

Magistritöö

Juhendaja
Sadok Ben Yahia
Professor
kaasjuhendaja
Imen Ben Sassi
Post-Doc Researcher

Tallinn 2021

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author:	Bilal Masud	*Bilal Masud*
		(signature)
Date:	May 01, 2021	

Annotatsioon

Asukohapõhiste sotsiaalvõrgustike, nagu Foursquare, Facebook ja mitmesugused veebi- ja veebirakendused, sealhulgas Netflix, YouTube, Spotify jne, populaarsus on kiiresti kasvanud. Koos populaarsusega on ka huvipunktide (POI) tähtsus muutunud otsustavaks. Raske on soovitada kasutajale huvitavaid objekte, mis võivad ka kasutajate tähelepanu köita miljonite objektide, nt asukohtade, laulude, toodete hulgast. Käesolevas väitekirjas teostame uuringu serendipity-põhise POI RS-i kontseptsiooni kohta, kasutades mõõdikuid relevantsus, ootamatus ja uudsus ning arutame ka serendipity mõju POI RS-is. Asjakohasuse puhul kontrollime kasutaja varasemate check-in'ide sarnasust teiste kasutajatega ja genereerime selle põhjal asjakohaseid nimekirju. Ootamatuse osas töötasime välja algoritmi, mis arvutab ka ootamatute asjakohaste objektide tõenäosuse. Samuti kujundasime algoritmi uudsuse mõõtmiseks, mis kasutab elemendi kontekstuaalset teavet ja määrab elemendi uudsuse. Et näha meie algoritmi tõhusust, kasutasime kahte tuntud andmekogumit ja tulemuslikkuse hindamiseks kasutasime 7 erinevat hindamismeetrit, sealhulgas f-measure, precision, recall, coverage, ILDGeo, diversity ja fidelity. Meie pakutud mudel on võrreldes tipptasemel mudelitega hästi toiminud.
Märksõnad: Soovitussüsteemid; Huvipunkt; Serendipity; Relevantsus; Ootamatus; Uudsus; Kontekstuaalne; f-measure; täpsus; tagasikutsumine; mitmekesisus; katvus; ILDGeo; truudus.

Abstract

The popularity of location-based Social Networks such as Foursquare, Facebook, and various web and online applications including Netflix, YouTube, Spotify, etc. has been booming. Along with the popularity, the importance of Point of interest (POI) has become crucial too. It is difficult to recommend user interesting items which can also catch users' attention from millions of items e.g. locations, songs, products. In this thesis, we use the concept of serendipity to create POI RS using metrics relevance and unexpectedness, and we also studied the impact of novelty on serendipity as well. For relevance, we check the commonality of user past check-ins with other users and generate relevant lists based on that. Regarding unexpectedness, we designed an algorithm that computed the probability of relevant items which are unexpected as well. We also design the algorithm for novelty metric, which uses the contextual information of the item and determines the novelty of the item. To see the effectiveness of our algorithm, we used two well-known datasets, and to evaluate performance, we used 7 different evaluation metrics including f-measure, precision, recall, coverage, ILDGeo, diversity, and fidelity. Our proposed model has performed well compared with state-of-the-art models.

Keywords: Recommender systems; Point of interest (POI); Serendipity; Relevance; Unexpectedness; Novelty; Contextual; F-measure; Precision; Recall; Diversity; Coverage; ILDGeo; Fidelity

List of abbreviations and terms

POI	Point of interest
RS	Recommendation system
LSBN	Location-based social network
IF	Information filtering
IR	information retrieval
CB	Content-based
CF	collaborative filtering
SAE	Self-attentive encoder
NAD	Neighbor-aware decoder
RBF	radial basis function
FPMC	Factorizing Personalized Markov Chain
TF	Tensor Factorization
RNN	Recurrent Neural Networks
STACP	Spatial Temporal Recurrent Neural Networks
CNN	Convolutional Neural Networks
RWR-KI	Random Walk with Restarts Enhanced by Knowledge Infusion
STACP	Spatio-Temporal Activity Center Point-of-interest
MF	Matrix Factorization
SMP	static model of user preference
TMP	temporal model of user preference
LGLMF	Local Geographical based Logistic Matrix Factorization
LGM	local geographical model
Rank-GeoFM	Ranking based Geographical Factorization Method
SOTA	State of the art

Table of Contents

List of Figures **vii**

List of Tables **viii**

1 Introduction **1**

1.1 Research Motivation . . . 2

1.2 Contributions . . . 3

1.3 Structure of the Thesis . . . 3

2 Basic Concepts **5**

2.1 Introduction . . . 5

2.2 POI-based Recommender Systems Overview . . . 5

2.2.1 Algorithms Classification . . . 6

2.2.2 Challenges . . . 9

2.2.3 Different POI Recommendation Systems . . . 9

2.3 Concept of Serendipity . . . 10

2.3.1 Serendipity definitions . . . 10

2.3.2 Serendipity and recommender systems . . . 11

2.4 Conclusion . . . 12

3 State of the Art **13**

3.1 Introduction . . . 13

3.2 POI Related Work . . . 13

3.3 Serendipity Related Work . . . 15

3.4 Overview of Important Models . . . 17

3.4.1 STACP . . . 17

3.4.2 LGLMFs . . . 19

3.4.3 Rank-GeoFM . . . 21

3.5 Conclusion . . . 24

4 A Serendipity-based Algorithm for POI Recommendation **25**

4.1 General Idea . . . 25

4.2 Serendipity-based Recommendation Approach . . . 25

4.3 Experimental Evaluation . . . 28

4.3.1 Experimental Setup . . . 28

4.3.2 Dataset Description . . . 30

4.3.3 Evaluation Metrics 30
4.3.4 Evaluation Results 32
4.4 Conclusion 42

5 Conclusion and Future Work **43**

Bibliography **44**

Appendices **49**

List of Figures

1 Classes of Recommendation Algorithms [9]. 6

2 Building the correlation matrix [39]. 16
3 User's spatio-temporal activity centers [46]. 18
4 Local geographical Model [37]. 20
5 Rank-GeoFM algorithm [49]. 23

6 Performance of our model for different values of parameter α [Tallinn dataset]. 33
7 Performance of our model for different values of parameter α [New York dataset]. 34
8 Comparison of diversity between Tallinn and New York dataset (left to right) 34
9 Comparison of Coverage between Tallinn and New York dataset (left to right) 35
10 Comparison of ILDGeo between Tallinn and New York dataset (left to right) 35
11 Comparison of Fidelity between Tallinn and New York dataset (left to right) 36
12 Comparison of Precision between Tallinn and New York dataset (left to right) 37
13 Comparison of recall between Tallinn and New York dataset (left to right) 37
14 Comparison of f-measure between Tallinn and New York dataset (left to right) 38
15 Comparison of diversity between Tallinn and New York dataset (left to right) 39
16 Comparison of coverage between Tallinn and New York dataset (left to right) 40
17 Comparison of ILDGeo between Tallinn and New York dataset (left to right) 40
18 Comparison of Fidelity between Tallinn and New York dataset (left to right) 41
19 Performance of our model by including the metrics (Relevance and Unexpectedness) and the metrics (Relevance, Unexpectedness, and novelty) for different values of parameter α 41

List of Tables

1 Collaborative filtering classes: Advantage and disadvantages [9]. 8

2 System used to run our models 29

1. Introduction

Thousands of items, including videos, music, books, and other utilities are available from a variety of online retailers. Because of the extensive selection of accessories, the store may have products that suit the user's tastes. However, having a vast range of items makes it harder for users to choose the product they choose to purchase. Thus, having more options does not necessarily mean that users will be satisfied [1].
Online stores use recommender systems (RS) to deal with such amounts of data. In this report, the recommender system (RS) refers to a software tool that offers relevant and unexpected items to the user. However, we will also see the impact of novelty along with relevance and unexpected metrics as well.
There are two types of recommendation systems: personalized and non-personalized systems [2]. Non-personalized RSs aim to recommend the same item to all users, while on the other hand personalized RSs aim to recommend different items based on the user's profile. In this report, we mainly focus on personalized RSs.
These latter give recommendations while considering the user's profile, which includes the information and actions users have performed before. Recommendations can be given base on:

- Ratings of the target user.
- Users' ratings provided on items attributes.
- Both items' attributes and users' rankings [3].

A RS may recommend items to users who have a lot of items in common with the target user. Another RS might suggest items that share a lot of characteristics with the target user's favorite items. One example will be if a user rates many horror movies, a movie RS will offer more horror movies to the target user.
RSs adopted for different purposes depending upon the goal itself. The goal of RSs differs based on each user's target aim [4]. For example, a point of interest (POI) recommendation system mainly considers the user's preference expressed based on his/her history of visits to generate recommendations.

1.1 Research Motivation

In recent years, the emergence of a vast number of Location-Based Social Networks (LBSNs), or online networks with location-based features, such as Foursquare, Flickr, Facebook, Google+, and others, has profoundly altered our vision of environment and how we communicate with it, thanks to the growth of Web 2.0 [5]. An LBSN is a form of online social network in which the content is linked to our external environment. As a result, our external environment and location have a significant effect on the framework of LBSNs, as well as the quality of services offered to consumers and how their personal data are processed. Indeed, LBSNs provide their users with a range of location-based applications, such as transportation, weather, news, and recommendation services. These services are particularly appealing to users who are dealing with a new or unfamiliar environment. Consequently, these networks have created an increasing number of innovations, technologies, and services to assist their users in exploring and discovering this new environment. Furthermore, since users have become familiar with connecting with web resources, a new knowledge necessity has emerged.

In reality, the amount of personal data and resources exchanged on these LBSNs has exploded in recent years [6]. On Flickr, for example, there are more than 112 million users who generate over 3.5 million images every day [7]. As a result of the information overload [8], users are finding it more difficult to locate what they are searching for in their surroundings. For instance, a consumer searching for a restaurant in a foreign country could become overwhelmed by the overwhelming amount of available data. Thus, over the last few years, a variety of POI search engines have been created to meet this need.

RSs have emerged as an important technology for addressing the issue of information overload. The main goal of RS is to offer assistance to users who need help to browse, rating, or sorting the huge amount of data available on LBSNs. These systems are also commonly used by online business channels in a broad range of contexts, including videos on YouTube, Vimeo, movies on Netflix, Amazon prime, music on Spotify, Soundcloud and POI can be found using applications like Foursquare. These sites are typically defined by the vast amounts of shared data they manage, i.e. 350 million users on Twitter send millions of messages every day, Amazon sells 350 million items, Spotify has 70 million songs, and Netflix has more than 4,000 movies [9, 10]. The user who wants to browse, scan or pick relevant online content faces severe functional limitations because of the large number of candidate items to investigate. Indeed, without an effective online assistant, navigating in these vast spaces becomes difficult. As a result, providing highly reliable recommendation lists and effective screening methods become a top priority in this context. Many RSs algorithms focus on the consistency that does not respond to the needs of users. High accuracy means that the RS has a higher power of prediction but may lack the ability to recommend unexpected and novel items. To gain consistency/accuracy, RSs recommend items relevant to the user's taste/profile [11, 12]. In this report, we are interested in serendipitous POI RSs which suggests relevant and unexpected items, but we will discuss the impact of novelty metric as well. Following are the challenges for generating serendipitous recommendations:

- There is no mutual agreement on the definition of serendipity in RSs and many definitions have been proposed by former approaches [11, 13].
- It is difficult to understand why users consider an item as serendipitous [14].
- Serendipitous items are not frequent as relevant items since they must be relevant, unexpected, and novel at the same time [11].

Since there is no mutual agreement on the definition of serendipity in RSs domain, in this work we are focusing on the following research questions:

- How can we define serendipity in the context of POI RS?
- How can it be used to improve the quality of POI recommendations?
- And how does serendipity affect the precision, recall, f-measure, coverage, diversity, and fidelity in the POI RS domain?

1.2 Contributions

We have designed a POI RS with a new approach using the concept of serendipity. For that, we have designed algorithms for the metrics relevance, unexpectedness, and novelty. After implementing these algorithms, we used accuracy metrics including precision, recall, and f-measure, as well as beyond accuracy metrics like the coverage, ILDGeo, diversity, and fidelity to evaluate the proposed RS.

1.3 Structure of the Thesis

Our thesis is structured as follows:

- **Introduction:** Our thesis is introduced in the first chapter. It details the research motivation and the major goals of our work. It also summarizes the contributions that we achieved throughout this thesis.
- **Basic Concepts:** The second chapter summarizes POI RS and its challenges and sheds light on different POI RSs. Then, it concludes by describing the concept of serendipity in the RS area.
- **State of the art:** The third chapter discusses the state-of-the-art of POI RSs. Then,

it describes serendipity related work. In the end, it summarizes the models we have used to compare our proposed system.

- **Approach:** This chapter describes our approach to serendipity in the context of POI RS. Then, it details the evaluation metrics we have used to measure the quality of our result that we compare with baseline models.
- **Conclusion and Future Work:** This chapter summarizes our work with detailed conclusions and discusses the future related work in serendipity-based POI RS.

2. Basic Concepts

2.1 Introduction

Foursquare and Facebook are two examples of location-based social networks (LBSNs) who have emerged recently. When people visit a point of interest (POI), including a gym, cafe, hotel, shopping center, or coffee shop, these web services enable them to check-in and express their interactions with friends. These networks are expanding at a rapid pace. Hence comes a need for a POI RS.
In this chapter we provide a POI RS overview, algorithms classification, some challenges in RS, different POI RSs. In the end it will discuss the concept of serendipity in the context of RSs.

2.2 POI-based Recommender Systems Overview

The primary goal of RSs is to provide users relevant suggestions [15]. Most of the cases, items are supposed to be brand new or at the very least, items that he/she will not locate on their own. These items are supposed to correspond to the user's taste and thus lead to a good user experience. That is why RSs use personalized analysis of a wide range of options. Uniquely in contrast to an information retrieval system in which the user explores the available space of options by straightforward queries, the RS does not know explicitly what the user truly needs. As an outcome, there is no straightforward query, the RS will only provide recommendations based on the user's previous interactions with the system. Due to this very reason RS gathers and examines all previous user's preferences to foresee future preferences.
Below are the definitions of terms and expression we used in the thesis:

Users: A user is someone who is registered on the LSBN. Each user have a profile described by the collection of all his/her previous check-ins.
Point-of-Interest: A point-of-interest is a uniquely identified site or location that is associated with a specific activity. In our thesis, a location is described by its latitude, longitude, and category tag. The terms point-of-interest, item and location are used interchangeably in this study to refer to the same thing.
Check-in: A check-in is the behavior of a user who interacted with (or visited) the POI.

2.2.1 Algorithms Classification

As described above, different items are recommended by personalized algorithms based on the specific user. It's possible that two users would get separate suggestions [3].
RSs are now commonly used for a variety of reasons. Increased turnover is the target of services which host RSs [16]. The methods for achieving the goal may vary depending on the implementation scenario. Since the hosting service's profit is dependent on user sales, one RS can recommend very cheap or costly items over interesting ones to a user. Another RS such as Telia or Elisa sell subscriptions, will recommend interesting items to the user so that user will visit again. Hence the goal of RS will vary according to the need of the business [4].
Filtering or retrieval techniques are often used to produce recommendations. The idea behind these techniques is to avoid unnecessary information in the aspect of information filtering (IF) or from databases in the case of information retrieval (IR)." Unwanted" information refers to the elements that are the least applicable to the target user in the context of recommendation. IF/IR-based suggestion methods take advantage of the presumption that user interests and desires are related. A user is expected to choose what other related users have selected in the past based on this premise. As a result, the most intuitive approach is to gather information about user interests and compare resemblances between user's profiles, then use the established interest of related users to create a prediction for the specific user. One of the most common classification emphases on the utilization of users, items, and system interaction and divides algorithms into three types: content-based, collaborative, and hybrid filtering, which will be discussed next. An hierarchy of recommendation algorithm is given in fig 1.

Content-Based Filtering: Content-based algorithms (CB) attempt to match a user's

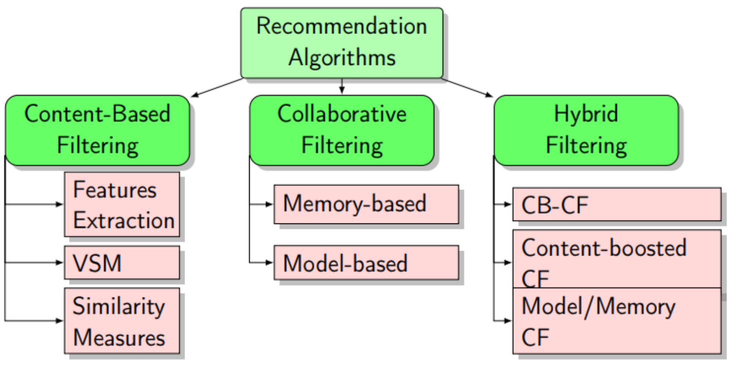


Figure 1. Classes of Recommendation Algorithms [9].

profile with items characteristics. This is accomplished in two stages. First, the model learns the tastes of the user's based on the user's previous history. This results in the

portrayal of a user's profile. After that it will rank the items that are most related to the items that the user previously enjoyed [17]. This requires having a common item representation for all items. This requires having a common item representation for all items. In the end, based on this ranked list, the model recommends unexperienced items. Item profiles and user profiles are typically described in CB filtering by a definition, such as a collection of keywords or attributes. When it comes to literary texts, for example, the keywords are the standard words of the vocabulary used inside the texts. Following the model's learning of each profile, the items are ranked using a similarity function.
Advantages and shortcomings: Since they are solely focused on content information and do not need any prior interaction history. There are two benefits of this [18]. The first CB approach ensures user independence by requiring no input from other users in order to make a recommendation. CB approaches do not suffer from the cold-start issue as a result of this: recommendations can be made also for new users or new items. Another benefit is that the system is more transparent: it's simple to understand why a decision was made based solely on other items rather than other users.
CB processes, on the other hand, have three significant drawbacks. They can only suggest items that are close to those that the consumer has already purchased. These approaches do not consider non-similar items and therefore appear to suggest the same types of items repeatedly, resulting in a lack of diversity but probably a lot of repetition. Another issue is the feature extraction. The features are easy to derive from textual data because they are simple and natural. This method, however, is not easily solved for complex data, for example, due to privacy concerns. Furthermore, CB methods are unable to differentiate between different items with the same features that might have different values for the user [9].
Collaborative Filtering: As compared to CB methods, collaborative filtering (CF) does not need item description. The word "collaborative" used here refers to how CF methods allow suggestions of items selected by the target user's most related users based solely on users' past encounters with the system [19]. The fundamental assumption is that users who have previously shown similar actions will continue to do so in the future as well. Therefore, since no extra detail, such as item description, is required, CF models are far easier to understand than CB models. This broadens the reach of CF methods and makes them more domain independent. Furthermore, since no personal information is needed, CF methods provide a higher degree of privacy. Another benefit of CF is that the more input the model gets, the more reliable the suggestion would be.
As shown in Table 1, collaborative approaches are classified as in two classes in literature [20] 1) memory-based 2) model-based. Memory-based methods specifically manipulate all the data in the user-rating matrix, while model-based methods only use a condensed version of the matrix. Within memory-based methods, neighborhood-based methods are the most often used: these methods take advantage of similarities between user and

item. Model-based systems are better customized, and for each user and item, it creates a compact model. This involves continuous access to the entire dataset to generate recommendations, which may pose significant problems as data volumes grow. Model-based methods avoid this issue because they only include a compact data model. One distinction is that neighborhood-based models are better suited to modeling local correlations, on the other side model-based approaches are better suited to modeling global relationships.

Table 1. Collaborative filtering classes: Advantage and disadvantages [9].

CF Class	**Techniques**	**Advantages**	**Shortcoming**
Memory-based	Neighbor-based Top-N depending on the user/item.	Fast and instinctive implementation. It is not necessary to create a new model when new data is available. On small datasets, provide fast recommendations.	When the amount of data is sparse, offer a low-quality recommendation. Since no user/item-content model has been developed, cold-start is an issue. Scalability problem.
Model-based	Clustering Methods Latent factors models Bayesian Networks Probabilistic Modeling	Handle the sparsity and scalability issues effectively. Improve the accuracy of prediction. Make recommendations more normal and intuitive.	The cost of creating a model is generally very high. Possible to lose any crucial information. A balance between prediction quality and scalability must be found.

Hybrid Filtering: Finally, this class combines algorithms from the other two classes. Based on finding that every model group has advantages and disadvantages; certain approaches aim to balance the benefits and the drawbacks of various strategies. As a result, hybrid models incorporate several suggestion approaches and consider both ratings and attributes of items.

2.2.2 Challenges

Typically, RS must deal with common issues relating to the consistency, quantity, anonymity, and protection of data. In the following section, we recommend a brief overview of these topics.

Sparsity: In general, the number of distinct items suggested by a RS is very large compared to the total number of registered users. Thus, even the most active user would be able to consume just a small portion of the available choices. It means that the user matrix density is very low: the density normally ranges from 0:005% to 1.5% [9]. It is a significant problem for a RS, which is supposed to make correct suggestions despite such bad input. This problem is called the sparsity problem.

Scalability: A RS is supposed to provide recommendations as quickly as possible in an environment where we are becoming more used to real-time communications and instant access to information. Since RSs are often associated with massive user-item databases (such as Alibaba, eBay, Facebook, YouTube etc.), they need a lot of computing power to generate recommendations in real time (a couple of milli-seconds maximum.). This demand needs the use of scalable approaches and effective data processing. Separating the learning process (which is performed offline) from the suggestion phase (online) through latent factors methods is a powerful solution.

Serendipity: Traditional RSs rely on users' previous behaviors and interests to recommend new items (e.g., books, places, etc.) to the users. As a result, there is a significant chance that the recommended items (or almost identical) may have been already appreciated by the user. When this phenomenon happens, it creates a problem called the serendipity problem. To improve the recommendations of the recommender systems, we should not only recommend new and relevant items, but we should endeavor to recommend significantly different items from what the user already has in their interest space. This phenomenon of surprising the user with a relevant but unique item is called serendipity. For example, a user likes the locations which are related to natural scenery. Among millions of locations, serendipitous POI RS will recommend the relevant and unexpected scenery locations. And a user may not discover those locations by himself/herself. in this report, we defined the serendipitous items to be relevant and unexpected to the users and but will also discuss the impact of novelty metric along with relevance and unexpectedness as well.

2.2.3 Different POI Recommendation Systems

Many recommendation services, for example user recommendation, task recommendation, and POI recommendation, are offered together in most of LBSN. POI recommendation is

one of the most difficult problems that has gotten a lot of attention from both academics and industry because of its business potential [9].
In the most common example, the POI RS receives a request that is solely based on the user's history. However, there are many more complex sub-problems that typically depend on side knowledge to accomplish a similar task. In the following section, we will go through related activities.
Next POI recommendation: The user's inquiry in this situation is often influenced by the user's current position. This problem's aim is to generate recommendations for the position and location of the current user. This means, the system will take the visit sequences into consideration [20]. Most current solutions to this issue, on the other hand, rely on techniques and methods from traditional POI recommendation.
POI Itinerary Recommendation: Many methods have been suggested for recommending a list of POIs that are subject to a time and/or financial budget. This is what the authors of [21] looked into: they added two strong restraints to the NP-hard optimal route problem to come up with a personalized approach. The authors in [22] suggest using a random walk approach to optimize user's touristic interactions between POI.
Time aware recommender system: The query that is obtained by the RS in this issue solely depends on the user's history. As with the previous challenge, the recommendation must consider how user expectations change over time. To model temporal control, the authors suggest a user-time-POI cube [23].

2.3 Concept of Serendipity

Serendipity is a complex phenomenon to research since it has an emotional dimension [24, 25]. Since serendipitous experiences are uncommon, it is difficult to describe serendipity in RSs, and what kinds of items are serendipitous and why [26].
Serendipity has long been regarded as one of the most difficult terms to translate. Serendipity is described as "the faculty of making fortunate discoveries by accident”, according to the dictionary [27]. The term "discovery" refers to the novelty of serendipitous experiences, whereas "fortunate" implies that the discovery should be relevant and unexpected.

2.3.1 Serendipity definitions

Here are some concepts in the relevant works on serendipity.

- Serendipity represents the "unusualness" or "surprise" of recommendations [28].
- Serendipity is the quality of being both "unexpected" and "useful" [29].

For two factors, increasing serendipity reduces precision in general. First, users are most likely to prefer things that are similar to what they already like, so many items are useless outside of their profiles. Second, considering the number of unpopular items [4] of poor quality, serendipity would probably lead to a rise in the number of items that are irrelevant/useless. As described before, in this report, we defined the serendipitous items to be relevant and unexpected to the users and but will also discuss the impact of novelty metric along with relevance and unexpectedness as well.

2.3.2 Serendipity and recommender systems

Various concepts of serendipity have been suggested in the RS. According to these concepts [30, 31], serendipity in recommender systems is described as the interestingness of items and the surprise experienced by users when they receive unexpected recommendations. As a result, in our thesis, we define serendipitous items by relevant and unexpected ones to the users, but will also discuss the impact of novelty on serendipity as well. Though, relevance is usually measured in terms of how well a recommendation matches the user's profile, determining the unexpectedness of a recommendation takes time. In the case of the location recommender system, relevant recommendations could be natural scenery locations which are similar to what the user has already visited but unexpected locations will be those which surprise or catch user attention and the user may not discover those locations on his/ her own.

To clarify the adopted meaning of serendipity, it is important to distinguish it from similar concepts such as novelty and diversity.

The novelty of a knowledge is how new/novel is from what a person or community has previously used. If a system that suggests an unfamiliar item, which a user might not find on his/her own, to the active user, it's considered a novelty [31].

The variety found in a list of recommendations is represented by diversity. Diversification methods are commonly used to eliminate homogeneous lists, on which all the recommended items are very similar to one another. If the user needs something different from the standard, this will lower the overall output of the recommendation list because none of the alternate recommendations would be liked. While diversity is not the same as serendipity, there is a connection between the two concepts in the sense that presenting a diverse list to the user will encourage unexpectedness [32].

2.4 Conclusion

The purpose of RSs is to provide useful recommendations. In the context of POI RSs, its approach has its own challenges and drawbacks. Different state-of-the-art methods will be presented in the next chapter.

3. State of the Art

3.1 Introduction

In this chapter, we start by discussing several start-of-the-art POI-related works in the following section. Next, we overview serendipity related work in sub-section 3.3. Finally, in sub-section 3.4, we summarize the baseline methods we selected to compare our results.

3.2 POI Related Work

The exponential growth of Location-based Social Networks (LBSNs), offers an excellent opportunity to meet the high requirement for personalized Point-of-Interest (POI) RSs. One such model SAE-NAD is proposed by [33], composed of a neighbor-aware decoder (NAD) and a self-attentive encoder (SAE), as a new auto-encoder-based model for learning nonlinear user-POI relations. Authors [33] proposed a neighbor-aware decoder (NAD) that uses the inner product of POI embeddings and the radial basis function (RBF) kernel to increase user's reachability on identical and nearby neighbors of checked-in POIs. Experiments have shown that this model outperforms several state-of-the-art baseline models.
Since it's difficult and important for recommendation systems to model user's diverse preferences based on their past behaviors. To make recommendations, previous approaches used sequential neural networks to encode user's experiences from left to right into hidden representations. Regardless of their efficacy, authors of this paper [34] claim that left-to-right unidirectional models are sub-optimal because of limitations such as a) The power of hidden representation is limited by unidirectional architecture in the user's action sequences b) Sometimes they assume a rigidly ordered, not necessarily realistic, sequence. They proposed BERT4Rec, a sequential recommendation model that models user activity sequences using deep bidirectional self-attention. They use the Cloze objective to the sequential recommendation to prevent information leakage and efficiently train the bidirectional model, predicting random masked items in the series by jointly conditioning on their left and right sense [34]. By having each item in past user behaviors merge information on both the right and the left, they learn a two-directional model of representation to make suggestions. After extensive testing, their model consistently outperforms various state-of-the-art sequential models.

Contextual knowledge, both spatial and temporal, is critical for understanding user behavior and prediction of user's action. With the increasing capacity to collect data, more temporal and spatial contextual data is collected in systems, making the location prediction issue more relevant and feasible. Authors of this paper [35] pointed out that some works suggested to solve this problem but each has its own constraints. They found out that Factorizing Personalized Markov Chain (FPMC) is built on the assumption of strong independence among different factors, which limits its performance. In predicting future actions, Tensor Factorization (TF) faces the cold start problem. When compared to PFMC and TF, the Recurrent Neural Networks (RNN) model performs well, but both approaches struggle to model continuous-time intervals and geographic distance. They extend RNN and propose Spatial-Temporal Recurrent Neural Networks as a fresh approach (ST-RNN) [35]. With time-specific transients in each layer, ST-RNN can model local temporal and spatial contexts. ST-RNN outperforms state-of-the-art approaches and can model spatial and temporal contexts well, according to experimental findings on real datasets.

In recent years, the focus of recommender system algorithm research has moved away from matrix factorization techniques and their latent factor models and toward neural approaches. However, some newer neural approaches integrate latent factor models into more complex network architectures, owing to their proven strength. Authors in this paper [36] mention that several researchers recently proposed a specific idea: consider potential correlations between latent factors, i.e., embeddings, by applying convolutions over the user-item interaction map. Contrary to what they stated in these articles, such interaction maps do not share the properties of images where Convolutional Neural Networks (CNNs) are particularly useful. Authors showed in [36] that the asserted improvements stated in the literature cannot be attributed to CNNs' ability to model embedding correlations, as argued in the original articles, through analytical considerations and empirical evaluations. Besi, additional performance evaluations show that all the recently examined CNN-based models outperform existing non-neural machine learning techniques or traditional nearest-neighbor approaches. On a broader level, their findings highlight major methodological issues in recommender system research.

As mentioned at the start, the Points of Interest (POIs) recommendation has become a vital challenge in assisting users in exploring their area/environment. Given the absence of check-in data, geographical data provides an incentive to increase the accuracy of POI recommendations. In [37], the authors mentioned matrix factorization methods which provide efficient models for POI recommendation. However, in order to increase the accuracy of POI advisory processes, two major issues must be tackled. First, using spatial data to capture both the user's own geographic profile and the geographic popularity of a site.

The second step is to use the geographical model in the matrix factorization methods. To resolve these issues, this paper [37] proposes a POI recommendation approach focused on a Local Geographical Model that takes into account both users' and locations' perspectives. To that end, the authors suggested a geographical model that takes into account the user's primary area of activity as well as the importance of and position within that area. The suggested local geographical model is then merged with the Logistic Matrix Factorization to increase POI recommendation accuracy. The suggested method outperforms other state-of-the-art POI recommendation approaches, according to experimental findings.

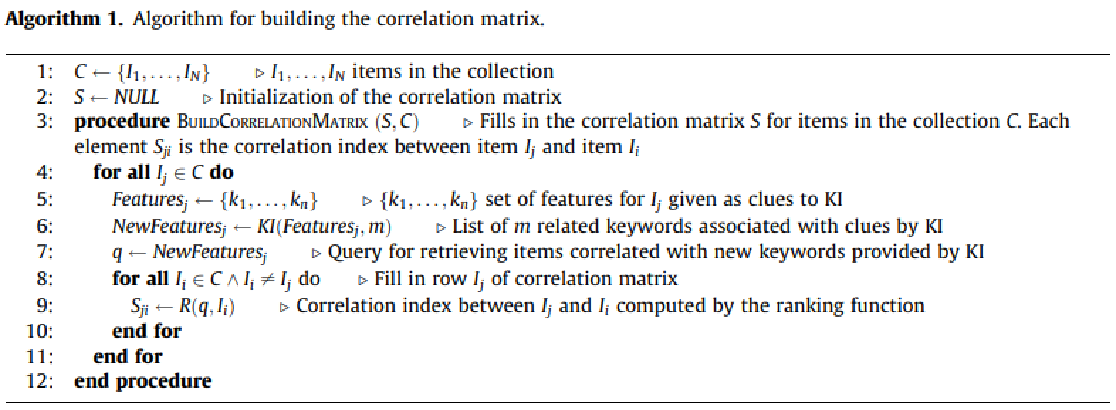
3.3 Serendipity Related Work

Graph-based techniques are gaining popularity because they allow for the capturing of transitive associations between nodes (items), facilitating the discovery of correlations [38]. The authors in[39] used a Random Walk model [40] called Random Walk with Restarts (RWR) as a recommendation technique for finding serendipitous items, which was improved by KI. Random Walk with Restarts Enhanced by Knowledge Infusion was the name given to the resulting algorithm (RWR-KI). To predict user preferences, Random Walk models use a correlation graph between items. The edges of the correlation graph show the degree of correlation between items, while the nodes correspond to the items. Fill in each entry with the correlation index between item pairs to create a correlation matrix. The correlation index in [41] denotes the number of users who co-rated the item pair, while the correlation index in [42] refers to the similarity of content between items. In the random walk model, given a correlation graph and a starting point, such as a user-favorite item, a neighbor of the starting point is randomly selected for a transition; then, a neighbor of this point is recursively selected at random for a new transition. There is a chance of returning to the starting node at each stage.

Building the correlation matrix using knowledge infusion: The idea for I is to use KI's keywords to measure the index of similarity between I and other items within the collection. The authors used a content-based model where each item I in n-dimensional function space is interpreted as a vector [43].
The vocabulary of the item set is the function space, which is composed of keywords derived from item descriptions. For a query q, a ranking-based function is adapted, which is based on a probabilistic retrieval system [44].

The process (the algorithm is shown in fig 2), for constructing the correlation matrix is divided into three steps:

1. For item I_j (step 5), choose the most representative features (keywords).



Algorithm 1. Algorithm for building the correlation matrix.

```
 1: $C \leftarrow \{I_1, \ldots, I_N\}$    ▷ $I_1, \ldots, I_N$ items in the collection
 2: $S \leftarrow NULL$    ▷ Initialization of the correlation matrix
 3: procedure BuildCorrelationMatrix $(S, C)$    ▷ Fills in the correlation matrix $S$ for items in the collection $C$. Each
    element $S_{ji}$ is the correlation index between item $I_j$ and item $I_i$
 4:   for all $I_j \in C$ do
 5:     $Features_j \leftarrow \{k_1, \ldots, k_n\}$    ▷ $\{k_1, \ldots, k_n\}$ set of features for $I_j$ given as clues to KI
 6:     $NewFeatures_j \leftarrow KI(Features_j, m)$    ▷ List of $m$ related keywords associated with clues by KI
 7:     $q \leftarrow NewFeatures_j$    ▷ Query for retrieving items correlated with new keywords provided by KI
 8:     for all $I_i \in C \wedge I_i \neq I_j$ do    ▷ Fill in row $I_j$ of correlation matrix
 9:       $S_{ji} \leftarrow R(q, I_i)$    ▷ Correlation index between $I_j$ and $I_i$ computed by the ranking function
10:     end for
11:   end for
12: end procedure
```

Figure 2. Building the correlation matrix [39].

2. Using KI to get new keywords linked to I_j by supplying such keywords as hints (step 6).
3. Ij-related items are retrieved with the KI providing new keywords to enter the ranking function (steps 7–10).

The correlation matrix is filled in using the scores calculated by the ranking function. For the evaluation, the authors have defined and used the metrics as below:
Relevance: If the given rating by user u on item i is higher than the average value of all ratings provided by u, then item i applies to a user u [39]. The following metric, as shown below, describes the importance of the recommendation list L of size N, as the ratio between the size of the subset of L that includes related objects and the size of L [39].

$$Relevance@N = \frac{\sum_{i \in L} R(i)}{N} \tag{3.1}$$

$$R(i) = \begin{cases} 1 & \text{if i is relevant} \\ 0 & otherwise \end{cases} \tag{3.2}$$

Unexpectedness: on the other hand, can be characterized independently of the user using certain common prediction criteria [45] such as popularity and item average ranking.
The ratio between the number of users who ranked i and the total number of users in the dataset is used to determine the item's popularity. If the item i's popularity score is lower than the average popularity measured over all items in the dataset, it is unexpected, according to this criterion. The other criteria consider the ratings that have been given to each item. The average user rating of Item i in the data set is the mean rating of the item. Under this definition, the short head contains items with an average rating higher than the

average rating calculated for all items in the dataset, whilst the long tail includes items with an average rating lower than the average, i.e. items that are less preferred by users and are most often unexpected. They discovered that 69% of items are unexpected while using popularity, while the number of unexpected items drops to 44% when using average rating [39]. If a criterion dependent on the rating is used, then this ensures it will be more difficult to recommend unexpected items.
We define the unexpected metric *N* of size *L*, as the ratio of the size of the subset of *L* that includes only unexpected items to the size of *L* [39].

$$Unexpectedness@N = \frac{\sum_{i \in L} U(i)}{N} \tag{3.3}$$

$$U(i) = \begin{cases} 1 & \text{if i is relevant} \\ 0 & otherwise \end{cases} \tag{3.4}$$

Serendipity: The serendipity metric *N* of size *L* is defined below, as the ratio of the subset size *L* that includes serendipitous items, those items will be both relevant and unexpected at the same time [39].

$$Serendipity@N = \frac{\sum_{i \in L} S(i)}{N} \tag{3.5}$$

$$S(i) = \begin{cases} 1 & \text{if i is relevant} \\ 0 & otherwise \end{cases} \tag{3.6}$$

The serendipity metric will contain both relevant and unexpected items. Authors of this paper [39] use serendipity to evaluable whether the recommendation list is serendipitous. However, to the best of our knowledge and research, we didn't find any approach which uses the concept of serendipity to generate recommendations.

3.4 Overview of Important Models

This section will give an overview of different baseline models we used to compare our results.

3.4.1 STACP

Authors [46] proposed a model STACP, a Spatio-Temporal Activity Center POI recommendation model that incorporates both user preferences and context. They build two preference functions for each user in the user's preference model to account for both static and temporal preferences. Furthermore, the influence of geographical and temporal

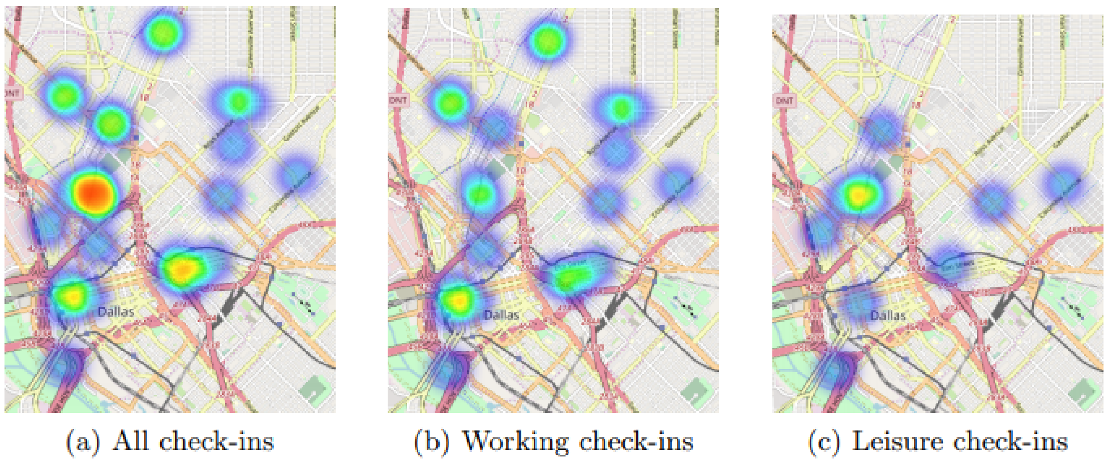


Figure 3. User's spatio-temporal activity centers [46].

knowledge is combined in the user's context model.
Let the set of users be $U = u_1, u_2, ..., u_m$, and the set of POIs be $L = l_1, l_2, ..., ln$. Let m and n represent the number of users and points of interest, respectively. The user's visit-frequency can then be encoded in $P(m*n)$, with entries $R_{u,l} \in P$ representing user $u \in U$'s previous POI check-ins to POI $l \in L$ [46].

Spatio-Temporal Activity Centers

As shown in fig 3 that Users' actions are centered, these centers vary depending on the temporal information's periodicity (see Figures 3 (b) and 3 (c)). [46] mentioned that this phenomenon highlights a flaw in previous geographical and temporal models, which failed to account for both geographical and temporal influences. The second feature of user behavior, as seen in fig 3, is that users prefer to visit POIs that are close to their present locations. They suggested the Spatio-temporal activity-centers model by combining these two characteristics to model users' check-in behavior. After that, given the temporal multi-center set $C_{u,t}$ of user u in time t, it will calculate the score of user u, visiting a POI l. The multiplication of two terms is used to calculate each center. In the first term, the score of the POI l that belongs to the center $C_{u,t}$ is determined by the distance between the POI l and the center $C_{u,t}$. The effect of check-in frequency $C_{u,t}$ on the center $C_{u,t}$ is denoted by the second term.
After that, the multi-center activity function is defined as a linear interpolation applied to two temporal states: working time and leisure time. We can describe other temporal states using the model. It may use it to model weekday, weekend, weekly, or regular trends.

Activity Center Allocation

The users' behaviors, as previously mentioned, follow a center-based pattern. Depending on the temporal state, these centers vary. They proposed a temporal multi-center clustering algorithm among each user's check-ins based on the Pareto principle [47] to model users' actions in a Spatio-temporal manner since the most visited POIs only account for a few users. First, it ranks all POIs L_u by check-in frequency for each user u and temporal state t. Then, to establish an area, it chooses the most visited POI and combines it with all other visited POIs within d kilometers of it. Let N_u represent the total number of check-ins for user u, r represent the new region, and $N_{r,u}$ represent the total number of check-ins for user u's current area. It uses a threshold of α to determine whether a center should apply to a user's profile. If $(N_{r,u}/N_u) > \alpha$, then a new center is proposed. This process is repeated until all the user u's checked-in POIs have been covered.

Static and Temporal User's Preferences

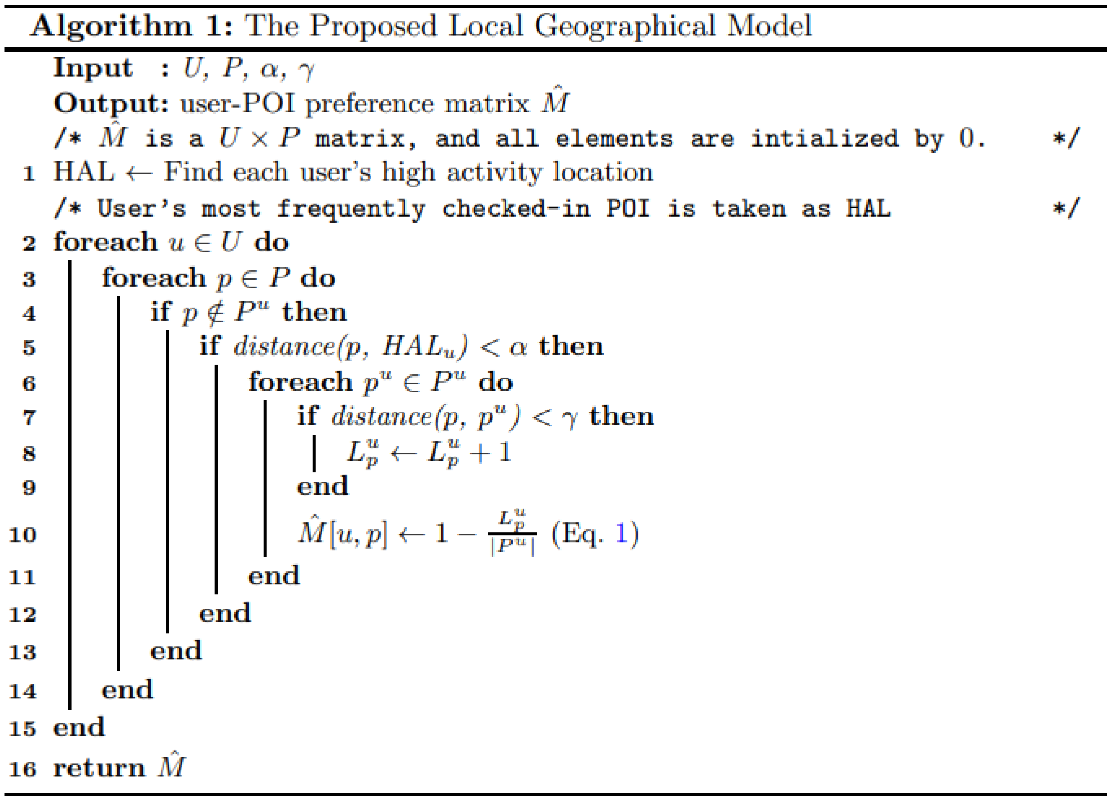
They used Matrix Factorization (MF) in two ways to model the user's preferences based on check-in data: a static model of a user preference (SMP) and a temporal model of a user preference (TMP). To model the static behavior of users, SMP uses the conventional matrix factorization model. Because of frequency matrix R, two low-rank metrics are the objective of MF-based recommendation. Then, similar to a POI l, it calculated a predicted recommendation score for a user u
To model users' temporal behavior in TMP, in accordance with the various time states T, the initial user POI frequency matrix R was subdivided into t sub-matrices, as inspired by [48]. Then, only check-in activity occurring at the corresponding time status is used with every submatrix. For instance, in this [46], they used $t = 2$ to represent working time and leisure time.

3.4.2 LGLMFs

Local Geographical based Logistic Matrix Factorization is the proposed POI recommendation process (LGLMF) [37]. LGLMF is divided into two phases. A Local Geographical Model (LGM) is being proposed in the first phase, focused on users as well as on locations. They then merged the LGM with a Logistic Matrix Factorization (LMF) approach in the second phase. To predict the preferences of users, the factorization model of the fused matrix is used.

Local Geographical Model

Let the set of users be $U = \ u_1, u_2 \ldots, u_m$, and the set of POIs $P = \ p_1, p_2, \ldots, p_n$. Then, with m users and n POIs, let C be a user-POI check-in frequency matrix. The value $C_{u,p}$ C displays the user u's check-in frequency at the POI p. Following that, the problem of personalized top-N POI recommendation is formally described as follows.
Given a user-POI check-in frequency matrix C and a set of POIs P that have been visited



```
Algorithm 1: The Proposed Local Geographical Model
   Input  : $U$, $P$, $\alpha$, $\gamma$
   Output: user-POI preference matrix $\hat{M}$
   /* $\hat{M}$ is a $U \times P$ matrix, and all elements are intialized by 0.   */
1  HAL ← Find each user's high activity location
   /* User's most frequently checked-in POI is taken as HAL                 */
2  foreach $u \in U$ do
3      foreach $p \in P$ do
4          if $p \notin P^u$ then
5              if $distance(p, HAL_u) < \alpha$ then
6                  foreach $p^u \in P^u$ do
7                      if $distance(p, p^u) < \gamma$ then
8                          $L_p^u \leftarrow L_p^u + 1$
9                      end
10                     $\hat{M}[u,p] \leftarrow 1 - \frac{L_p^u}{|P^u|}$ (Eq. 1)
11                 end
12             end
13         end
14     end
15 end
16 return $\hat{M}$
```

Figure 4. Local geographical Model [37].

by the user u, identify X = $p_1, p-2, ..., p_n$, a set of POIs ordered based on the probability of a user's visit in the future such that $|X| \leqslant N$ and $X \cap P^u = \phi$.

The proposed spatial model accounts for both user and location geographical influences. We can model geographic details from the user's perspective by considering their activity area. From the position point of view, the number of check-ins made in the neighbor of a selected POI can be calculated as geographical information. As a result, a location's agreeability in relation to its neighbors can be determined. The proposed LGM's pseudocode is shown in fig 4. The algorithm is made up of three internal loops that model geographical data. In both the first and second loop, the area of a user's model from (lines 2–5), and the third loop, taking into consideration neighborhood POI visits (lines 6–10), compute the chance of a user preferring POIs [37]. It will find each user's high activity location (which may be the user's residence area) in order to model the user's area. From the user's perspective (user's high activity region), it will look for unvisited POIs near that user, and α shows the range in kilometers line (5). Furthermore, for each user based on in-region POI, the effect of the checked-in neighboring POIs whose distance from unvisited POIs is less than γ meters (location's perspective) will be considered (lines 7-10). Line 10 represents the locality of POI, where Lup stands for the number of u visited neighbor p and $|P^u|$ represents the length of POIs user u has visited.

Constructing the Matrix Factorization Model

Considering the existing local geographical model (LGM) as additional contextual details, a novel matrix factorization approach based on logistic Matrix factorization (LMF) is proposed. Two low-rank matrices found by the MF-based recommendation, include the user-factors matrix V and the item-factors matrix *L*, whose inner product approximates matrix C. Each V row represents a user's vector of behaviors, while each L row represents the act of item properties.
Assume that $e_{u,p}$ is the number of times user *u* has checked in on POI *p* (user *u* prefers POI *p*), and that *V* and *L* are two latent variables for users and POIs, respectively. Then, it calculates the probability that the user *u* preference on the POI *p*.
Finally, the proposed LGM is integrated into the matrix factorization model which will calculate the probability of user *u* who will visit POI p. Using the proposed probability function, we can give a list of POI recommendations for each user. It's worth noting that, unlike the LMF approach, the proposed LGLMF model incorporates contextual data into the recommendation process by combining the proposed LGM [37].

3.4.3 Rank-GeoFM

The proposed ranking-based factorization approach for POI recommendation is presented in this method [49]. First, it will introduce how to optimize the POI recommendation problem by first planning it as a rating objective function. Finally, it applies the model to the recommendation of time-aware POIs.

Ranking Based Geographical Factorization

Preference Ranking Objective Function: it will begin by defining an aim function for POI recommendation. Because of the sparsity of check-in data, authors have designed the aim function by fitting the user's preference rankings for POIs, rather than fitting user check-in frequencies, as conventional factorization methods do. [49] believe that the higher the check-in frequency, the more a user prefers the POI; and that unvisited POIs are less favored than those that have been visited. They developed a method to calculate the incompatibility between the inferred rankings and the rankings generated by a factorization model based on their intuition. They calculated the incompatibility in particular for a given user u and POI l. After that, it counts the number of POIs that, according to the check-in results, should be ranked lower than l for user u, but are ranked higher by the factorization model. The number of POIs that are incorrectly ranked higher than l for user u is then measured, and this is referred to as "ranking incompatibility" [49].
Then, for learning a factorization model, they designed a preference ranking objective function. They suggested the objective function for minimizing, which requires a function

to turn the ranking incompatibility with a loss since a successful factorization approach can minimize the ranking incompatibility as often as possible. It will add up all the losses for all the user-POI pairs to calculate the total loss. The number of losses at each rank point (from 1 to r) for the incorrectly-ranked POIs is then calculated, with each position i was given a loss $1/i$. One advantage of this objective function is that it can deal with data sparsity. All the other POIs, which are often unvisited POIs, decide the rating incompatibility of a POI l for a consumer (because a user often visits very few POIs). As a result, although they are overlooked in traditional MF, the unvisited POIs contribute to learning the model. As a result, it solves the sparsity issue of check-in data by using the objective function rather than directly fitting zero check-ins.

Geographical Factorization Model: This subsection will explain a method that is used for calculating the recommendation score using geographical factorization. This factorization model can identify a user's interests through POIs. Furthermore, it considers the impact of the geographical background on POI recommendations. On the one hand, they use matrices to parametrize the latent factors of users and POIs in a dimensional space. They used these in the same way as conventional matrix factorization approaches are used to model the user's own preferences. It implements an additional latent factor matrix and uses it to model the interaction between users and POIs in order to account for geographic influence. It considers per POI l k-nearest neighbors. It assumes that users would visit nearby POIs. Since each row of the matrix reflects the effect probabilities, it will be normalized.

Assume that the parameters of the geographical factorization model have already been learned. It calculates a recommendation score for user u and POI l. Then it calculates the user preference score and also calculates the geographical impact score, which indicates how much a user likes a POI because of its surroundings. It constrains the latent factors of our model into a ball, which acts as a regularizer [50], to avoid the over-fitting problem. They constrained the latent factors of different parameters into a small ball with radius C and αC. To balance the contributions of user preference and geographical effect ratings, it implements the hyper-parameter α. As a result, balancing the contributions of user-preference and geographical effect scores to the final recommendation score can be achieved by tuning the hyperparameter.

The proposed Ranking based Geographical Factorization Method (Rank-GeoFM) is shown in fig 5. It iterates through all the user-POI check-in pairs in the algorithm, updating the latent factors until the procedure converges (lines 3-16). The sampling process is first performed in each iteration, given a user-POI pair, in order to estimate ranking incompatibility and get one POI sample (lines 6-8). Using the stochastic gradient descent (SGD) method, it updates the related latent factors based on the estimate of ranking incompatibility and the sampled POI (lines 9-15). The revised latent factors are checked for norm constraints, and those that violate the constraints are predicted (line 16).

```
Algorithm 1: Rank-GeoFM
input : check-in data $\mathcal{D}_1$, geographical influence matrix $\mathbf{W}$,
        hyperparameters $\varepsilon$, $C$ and $\alpha$, and learning rate $\gamma$
output: model parameters $\Theta = \{\mathbf{U}^{(1)}, \mathbf{L}^{(1)}, \mathbf{U}^{(2)}\}$
1  Initialize $\Theta$ with Normal distribution $\mathcal{N}(0, 0.01)$;
2  Shuffle the samples in $\mathcal{D}_1$ randomly;
3  repeat
4      for $(u, \ell) \in \mathcal{D}_1$ do
5          Compute $y_{u\ell}$ as Eq. (4), and set $n$=0;
6          repeat
7              Sample a POI $\ell'$;
8              Compute $y_{u\ell'}$ as Eq. (4), and set $n = n + 1$;
           until $I(x_{u\ell} > x_{u\ell'})I(y_{u\ell} < y_{u\ell'} + \varepsilon) = 1$ or $n > |\mathcal{L}|$;
9          if $I(x_{u\ell} > x_{u\ell'})I(y_{u\ell} < y_{u\ell'} + \varepsilon) = 1$ then
10             $\eta = E\left(\left\lfloor \frac{|\mathcal{L}|}{n} \right\rfloor\right) \delta_{u\ell\ell'}$;
11             $\mathbf{g} = \left( \sum_{\ell^* \in \mathcal{N}_k(\ell')} w_{\ell,\ell^*} \mathbf{l}_{\ell^*}^{(1)} - \sum_{\ell^+ \in \mathcal{N}_k(\ell)} w_{\ell,\ell^+} \mathbf{l}_{\ell^+}^{(1)} \right)$;
12             $\mathbf{u}_u^{(1)} \leftarrow \mathbf{u}_u^{(1)} - \gamma\eta(\mathbf{l}_{\ell'}^{(1)} - \mathbf{l}_\ell^{(1)})$;
13             $\mathbf{u}_u^{(2)} \leftarrow \mathbf{u}_u^{(2)} - \gamma\eta\mathbf{g}$;
14             $\mathbf{l}_{\ell'}^{(1)} \leftarrow \mathbf{l}_{\ell'}^{(1)} - \gamma\eta\mathbf{u}_u^{(1)}$;
15             $\mathbf{l}_\ell^{(1)} \leftarrow \mathbf{l}_\ell^{(1)} + \gamma\eta\mathbf{u}_u^{(1)}$;
16         Project the updated latent factors to enforce constraints
           in Eqs. (5)~(7), e.g., if $\left\|\mathbf{u}_u^{(1)}\right\|_2 > C$, then set
           $\mathbf{u}_u^{(1)} \leftarrow C \frac{\mathbf{u}_u^{(1)}}{\left\|\mathbf{u}_u^{(1)}\right\|_2}$;
   until convergence;
17 return $\Theta = \{\mathbf{U}^{(1)}, \mathbf{L}^{(1)}, \mathbf{U}^{(2)}\}$
```

Figure 5. Rank-GeoFM algorithm [49].

Time-aware POI Recommendations: This section shows how to easily generalize this approach to include different types of contexts. It considers time-aware POI recommendations.

It extends two additional terms to capture the temporal component in order to measure the recommendation score. One term is temporal popularity score, which shows whether this POI is common during the specified time period. The other term is the temporal effect score. Thence, the popularity of a POI during a onetime slot is often affected by other time slots that are nearby. Aside from the two additional concepts, it implements three more latent factor matrices. It parameterized the latent factors of time slots as a matrix T, as well as two additional latent factor matrices L2 and L3 for POIs, where L2 is used to model interactions with time slots for temporal popularity score, and L3 is used to model interactions with near or related time slots for temporal influence score. It also creates a matrix *M* and normalizes *M* into a matrix with each row representing a probability vector. They calculate the recommendation score of POI l given a user u and a time slot t.

To prevent over-fitting, norm constraints are applied, similar to the POI-user environment. The algorithm, as shown in fig 5, can easily be modified to suggest POIs that are time-aware. To update the latent factors, it iterates over all user-time-POI tuples.

3.5 Conclusion

There are different state of the art models presented in this chapter. For example, a novel model STACP which jointly models the geographical and temporal influence of the user's check-in behavior along with user's temporal information and user's preference on POIs and it works well in large datasets. In contrast, a novel LGLMF model considers both users and locations point of view of geographical information. We also discuss the ranking based factorization method called Rank-GeoFM which learns the factorization by fitting the users preference ranking for POIs and also how this algorithm can be generalized for time-aware POI RS.
This thesis presents a novel serendipity-based POI RS algorithm to give user's relevant and unexpected recommendations and to evaluate our model we used 7 different metrics. A detailed introduction of our proposed model is given in the following section.

4. A Serendipity-based Algorithm for POI Recommendation

4.1 General Idea

In this chapter, we propose a novel Serendipity-based point of interest recommendation system which aims to recommend relevant and unexpected POI. For this, we design two metrics, relevance, and unexpectedness. For each user, the system measures the similarity among other users who have similar tastes and extract uncommon POIs, which in this case apply to the user. As for the unexpectedness, for each user, it considers its past and relevant POIs and measures the probability of which we rarely visited together POIs. We consider the remaining POIs serendipitous to the user. However, to analyze the impact of the novelty component, we include it in the serendipity definition. We compare the got results based on the 2 components serendipity algorithm and the three components one. We discuss all the details about our approach in this chapter.

4.2 Serendipity-based Recommendation Approach

In this section, a detailed approach to the serendipity-based recommendation is discussed with metrics Relevance, Unexpectedness, and Novelty.
First, we split the dataset into a training and testing set with a ratio of 70 and 30%. Then make a pair list of users and his/her visited locations. After that, we used defined relevance metrics.
Relevance: Serendipity can be assessed in a variety of ways. As for relevance, one of the serendipity's components. In this paper, we define relevance in the Algorithm 1.

The Relevance algorithm, as shown below, starts by using each user in the test set and checks if user u exists in the training set (line 2) then extracts its visited POIs line (line 3). After that, if loop trough for each user, locations pair (k, visited locations) in the training set (line 4) and put constraint if both users are not same (line 5) then extract the common and uncommon locations between them and if the length of common locations $>= \alpha$ then if will recommend all the uncommon locations to that user u (line 6-9). Based on the factor α, the relevance list will vary.

Algorithm 1: Relevance

```
1- for each user u in test_set do
    2- if u in training_set then
        3- Extract visited_locations of u
        4- for k, l in training_pair do
            5- if if k != u then
                6- Extract Common_Locs of user u and k
                7- Extract uncommon_locs of user k
                8- if len(common_locs) >= $\alpha$ and len(uncommon_locs)!= 0 then
                    9- Relevant_List of locations for each user
                end
            end
        end
    end
end
```

Unexpectedness: Serendipity, according to Kaminskas and Bridge, has two components: unexpectedness and significance [51]. Unprecedentedness, according to the authors [38], represents how dissimilar a suggested item is to a user profile. Kaminskas and Bridge proposed two pair-wise similarity measures to quantify unexpectedness: point-wise mutual knowledge and content-based similarity. Based on the number of users, who have visited both items and each item separately, point-wise mutual knowledge, defined in Equation 4.1, shows how similar two items are [51].

$$PMI(i,j) = \frac{log_2 \frac{p(i,j))}{p(i)p(j)}}{-log_2 p(i,j)} \tag{4.1}$$

Where $p(i)$ is the probability of visiting the item i, and $p(i,j)$ is the probability of visiting the items i and j, together. PMI value ranges from -1 to 1, where -1 shows that two items never visited together and 1 shows that two items i, j are visited together. Since we want the items to be unexpected, it will extract the items which are in the range $PMI >= -1$ and $PMI < 0$. Algorithm 2 is used for unexpectedness computation.
From line-1, it will take each user u in the test set and check if it exists in the training set and extracts visited locations of u and if that user u exists in the relevant list then extract its relevant list of locations line (1-5). For each location i in the relevant locations of u, it will check if i exists in the locations visited by users. Then, it will calculate the probability of location i denoted as $p(i)$ line (6-7). After that for each visited location j of user u, such that location i and j are not the same and j exists in the list of locations which are visited by users, it then calculates $p(j)$ and also the number of users who visited location i and j together line (8-11). If it is greater than 0 then it will calculate the $p(i,j)$ and its PMI

Algorithm 2: Unexpectedness

```
1- for each user u in test_set do
    2- if u in training_set then
        3- Extract visited$_{l}ocations\_of\_u$
        4- if u in Relevant_List then
            5- Extract relevant_locations_of_u
            6- for r i in relevant_locations do
                5- if i in each location_visited_by_users then
                    6- Extract number_of_users who visited i
                    7- calculate p(i)
                    8- for r j in visited_locations_of_u do
                        9- if $i! = j$ and j in each location_visited_by_users then
                            10- calculate p(j)
                            11- calculate users who visited i__j
                            12- if len(users who visited i__j)$! = 0$ then
                                13- calculate $p(i, j)$
                                14- calculate $PMI(i, j)$
                                15- if $PMI - 1$ and PMI <0 then
                                    16- tempList.append(u, i, j, PMI)
                                end
                            end
                        end
                    end
                    17- Relevance_Unexpected_List.extend(tempList)
                    18- templist.clear
                end
            end
        end
    end
end
```

value with the formula as shown in the equation 4.1 If the $PMI >= -1$ and $PMI < 0$ then it means that locations are never visited together (12-16). Then, it will find unexpected locations for each relevant location in the list (c.f, Lines 17-18). In the end, the resulting list will be relevant and unexpected at the same time.

Novelty: The novelty metric was suggested by Vargas and Castells [52]. The metric is based on a user's distance from previously consumed items (user profile) [51].

$$nov_{va}^{dist1}(i,u) = min_{j \in I_u} dist(i,j) \tag{4.2}$$

$$dist(i,j) = 1 - sim(i,j) \tag{4.3}$$

The distance between items i and j is denoted by $dist(i,j)$, as shown in equation 4.3, where $sim(i,j)$ denotes any similarity between i and j ($sim(i,j) \in [0,1]$). In our model, we use the word similarity using spaCy library based on category tags. spaCy model tells us how two words are close to one another, semantically. If items i and j have higher similarity close to 1, it means that the distance between i and j is less. So, for novelty, we put a constraint to check whether the distance $>= 0.5$, which gives us un-similar items. Algorithm 3 is used for the novelty computation.

In the Novelty Algorithm 3, it will take each user u in the test set and compare if it exists in the training set and in $Relevance_and_unexpected$ list then extracts the visited locations of u and $relevant_and_unexpected$ locations of u (lines 1-4). Then, for each location l in the $relevant\_and\_unexpected\_locations\_of\_u$, if l does not exist in the visited locations of u then for each loc c in the visited locations it will extract the tags for both location l and c (lines 5-10). After that, it will find the semantic similarity between two tags l and c using the spaCy library and then calculate the distance line (11-12). In the end, it will check if the distance $>= 0.5$. So, two locations are not similar, then we add the location l to the list (lines 13-14). Resulting recommendation list will be relevant, unexpected and novel at the same time to the user u.

4.3 Experimental Evaluation

4.3.1 Experimental Setup

The hardware and software environment of experiment are shown in Table 2

Algorithm 3: Novelty

```
1- for each user u in test_set do
    2- if u in training_set and u in Relevance__Unexpected List then
        3- Extract visited_locations of u
        4- Extract relevant_and_unexpected_locations for u
        5- for l in relevant and unexpected locations of u do
            6- if l not in visited_locations then
                7- for c in visited_locations do
                    8- if (l in location_tag_list) and (c in location_tag_list) then
                        9- Extract tag for Location_l
                        10- Extract tag for Location_c
                        11- find the similarity using spaCy lib between above
                          obtained two tags
                        12- calculate the distance = 1- similarity
                        13- if distance0.5 then
                            14- Relevance_Unexpected_&_novel_List.append(u, l)
                        end
                    end
                end
            end
        end
    end
end
```

Table 2. System used to run our models

Programming language	Python3.7
Software	Anaconda (Spyder)
Ram	8 GB
Processor number and frequency	i7-9750H 2.60 GHz-4.50 GHz

4.3.2 Dataset Description

We used two real-world datasets from Flickr Tallinn and New York Foursquare LBSNs. Tallinn dataset comprises $15,376$ check-ins made by $1,853$ users on 429 POIs with 13 categories and a sparsity of 98.06%. On the other hand, the New York dataset consists of $227,428$ check-ins made by $1,083$ users on $38,333$ POIs with 250 categories and sparsity of 99.45%. Every check-in contains a user-id, POI-id, time, geographical coordinates, and category tag. We split each dataset into 70% and 30% ratio, where the 70% is used for training purposes, while the remaining 30% is for testing.
As mentioned in chapter 1 that there is no mutual agreement on the definition of serendipity. So, in this thesis we used metrics relevance and unexpectedness to obtain our results, however, to see the impact of novelty along with relevance and unexpectedness, we also got results with novelty metric as well.

4.3.3 Evaluation Metrics

As for evaluation metrics, we used 7 metrics including Precision, Recall, F-measure, Coverage, ILDGeo, Diversity, and Fidelity.
Among all the recommended POIs we choose the best POIs based on the popularity of each POI defines as k best POIs. We calculated the results with the value of $k = \{5, 10, 15, 20\}$.
The number of positive recommended items divided by the number of actual items known as *recall*, while the number of positive recommended items divided by the number of recommended items known as *precision*.
The Precision is defined in Equation 4.4 [53].

$$Precision@k = \frac{1}{M}\sum_{u \in U} \frac{\left|Reclist_u^k \cap ActualList_u\right|}{\left|ActualList_u^k\right|} \tag{4.4}$$

The Recall is defined in Equation 4.5 [53].

$$Recall@k = \frac{1}{M}\sum_{u \in U} \frac{\left|Reclist_u^k \cap ActualList_u\right|}{\left|ActualList_u\right|} \tag{4.5}$$

The F-measure is defined in Equation 4.6 [53] as:

$$F - measure@k = 2 * \frac{Precision@k * Recall@k}{Precision@k * Recall@k} \tag{4.6}$$

The term "coverage" [54] refers to how well the system covers the entire set of POIs. As defined in Equation 4.7, the coverage is specified as the percentage of POIs that appear in all users' Recommendation lists (Reclist), where Reclist refers to user u's recommendation list of length *k*.

$$Coverage@k = \frac{\left|U_{u \in U} RecList_u^k\right|}{|POI|} \tag{4.7}$$

For each user, ILDGeo calculates the pair-wise dissimilarity of POIs in the recommendation list as defined in Equation 4.8 [55]. To calculate the geographical diversity, we measured the dissimilarity using Equation 4.9.

$$ILDGeo_u@k = \frac{\sum_{(i,j) \in RecList_u^k} dissim(i,j))}{\left|RecList_u^k\right| * (\left|RecList_u^k\right| - 1)} \tag{4.8}$$

$$dissim(i,j) = kmDistance(loc_i\ , loc_j) \tag{4.9}$$

where *kmDistance* is the kilometer distance between two POIs calculated using the longitude and the latitude.

Diverse categorical aspects are shown by DivCat (c.f. Equation 4.10). It counts the number of distinct categories included in each user's recommended list [55]. Since the POIs can be tagged as the same categories, then *Cat* is the category to which the POI belongs.

$$DivCat_u@k = \left|U_{i \in\ RecList_u^k} Cat_i\right| \tag{4.10}$$

To test the recommendation explainability, the fidelity measure (c.f. Equation 4.11) [56] is defined as the percentage of explainable items in the recommended list as defined in

Equation 4.12 [55].

$$Fidelity_u = \frac{\left|RecList_u^k \cap Explainable_u\right|}{\left|RecList_u^k\right|} \tag{4.11}$$

$$Explainable_u = U_{v \in Neighbor_u} Neighbor_v \tag{4.12}$$

Where neighbor u refers to other users who have visited the most common POI. We set the value of neighbors, in our case the common user locations denoted as α, differently for each dataset which is shown in the result section. The term "visited" refers to all the points of interest (POIs) that v has visited. In our case, the relevance list for each user u is explainable. Hence, in our case, it accounted for max fidelity=1 each time.

4.3.4 Evaluation Results

To evaluate the performance of our model, we used 7 different metrics including f-measure, recall, precision, coverage, ILDGeo, diversity and fidelity. Three metrics f-measure, recall and precision, results are shown in the graphs below as $Result@k$, $k \in \{5, 10, 15, 20\}$. We conducted these results using the metrics Relevance and Unexpectedness. The effect of parameter α, which donates the representation of common locations among users, on the performance of our model is shown in the rest of this subsection.

Evaluation of Serendipitous Recommendation based on Relevance and Unexpectedness

In this subsection, we conducted our experiments on the results got by defining the serendipity using the relevance and the unexpectedness. Below are the results for the Tallinn dataset for different values of the parameter α.
Figure 6 shows the performance of our model by varying the parameter α, which defines the threshold fixed to extract the similar users to the target one based on their common locations. As described before, the precision measures the number of recommended items that belong to the actual list [53]. However, the recall measures the number of recommendations that are made out of all the actual list [53]. The F-measure of the system represents the harmonic mean of the precision and recall [57]. Initially, when the value of $\alpha = 1$ for $Result@k = 5$, the precision value equal to 0.0567 is considered low as compared to the recall value equal to 0.1164. Here, the f-measure value was equal to 0.0763. At $\alpha = 2$ the precision increased to reach 0.0726, the recall decrease to 0.1204,

and the f-measure has reached 0.0906. Our goal here is to find the optimal value of α which can be found via the f-measure to balance out both precision and recall. So, when we set the parameter of graph $Results@k = 5$ to $\alpha = 5$, we get the best value of the F-measure equal to 0.09917 with a precision of 0.1121 and a recall of 0.0888. The same is the case of $Results@10$, we get the best value when we set $\alpha = 5$ to get an f-measure of 0.1034, a precision equal to 0.0847, and a recall equal to 0.1325. For the graph $Result@15$, we obtained optimal values with an f-measure equal to 0.09907, a precision equal to 0.07155, and a recall value equal to 0.1610. At the last, for $Results@20$, when $\alpha = 5$, the f-measure was equal to 0.0903, when the precision and recall were equal to 0.0607 and 0.1759 respectively.

Figure 7 depicts the performance of our model by varying the value of α for the New

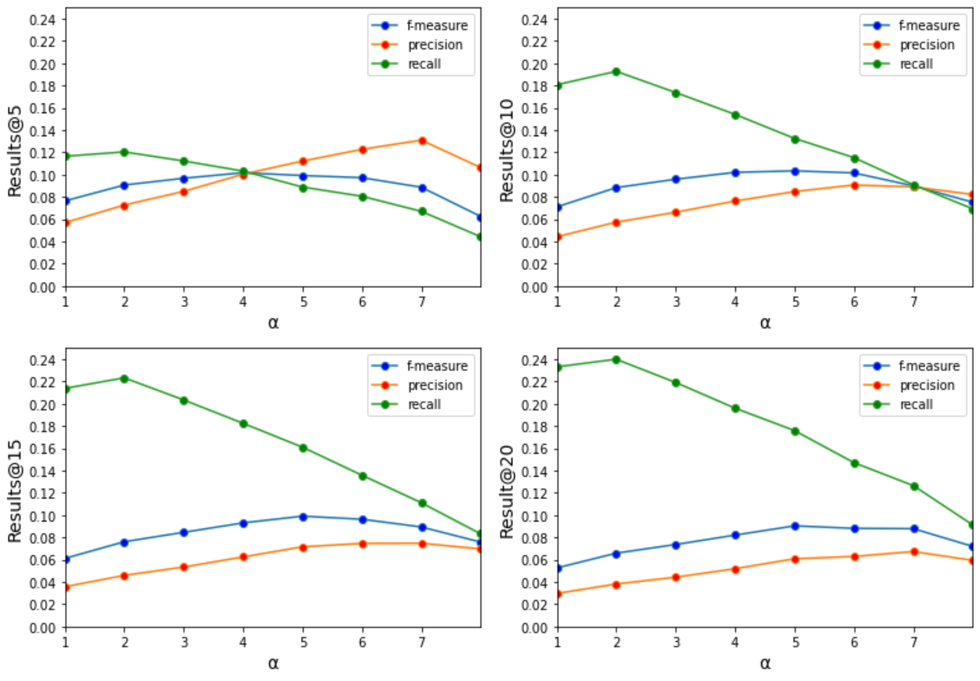


Figure 6. Performance of our model for different values of parameter α [Tallinn dataset].

York dataset. For the graph, $Results@5$ at $\alpha = 4$ the got precision is equal to 0.0311 which can be considered quite high as compared to both recall and f-measure equal to 0.0029 and 0.0054 respectively. Nevertheless, our goal is to balance the precision and recall metrics. By increasing the value of α the values of all the metrics are increasing as well and we got the best results at value $\alpha = 10$, with an f-measure equal to 0.0176, a precision of 0.0481, and a recall equal to 0.0108. For the graph $Results@10$, the optimal values are obtained also at $\alpha = 10$, with an f-measure, a precision, and a recall equal to 0.0196, 0.0383, and 0.01317 respectively. Equally, for the $Results@15$ and $Results@20$, we got the best accuracy values at $\alpha = 10$.

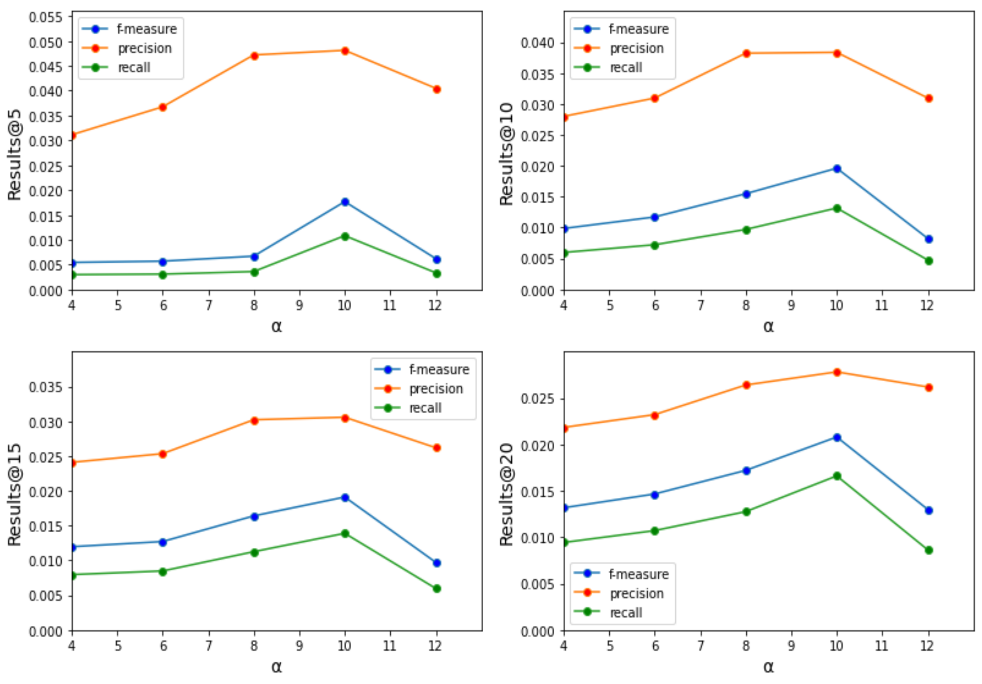


Figure 7. Performance of our model for different values of parameter α [New York dataset].

As we have obtained the optimal values of $\alpha = 5$ and $\alpha = 10$ for Tallinn and New York dataset respectively, we will use the same values on the rest of this chapter for the beyond accuracy evaluation.
The diversity represents how diverse the recommended items are in terms of their categories. This measure is calculated by having a union of all the unique categories in the recommendation lists. As we can see from figure 8, the value of diversity is increasing with the increase of the value of k. It means that when the recommendation list increases; it yields users more diverse items to explore.

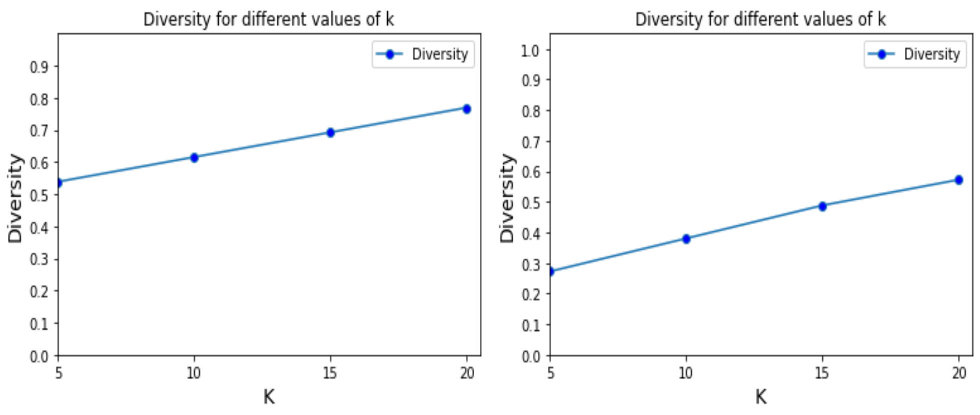


Figure 8. Comparison of diversity between Tallinn and New York dataset (left to right)

The coverage represents the total number of unique items in the whole recommendation list

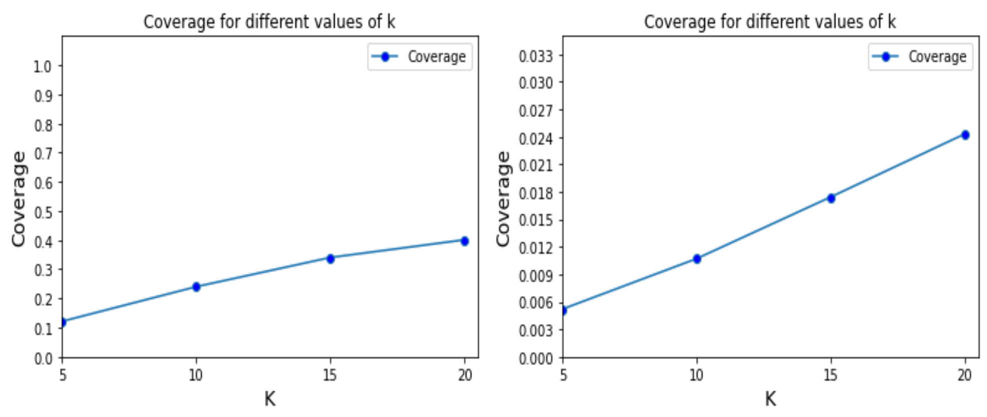


Figure 9. Comparison of Coverage between Tallinn and New York dataset (left to right)

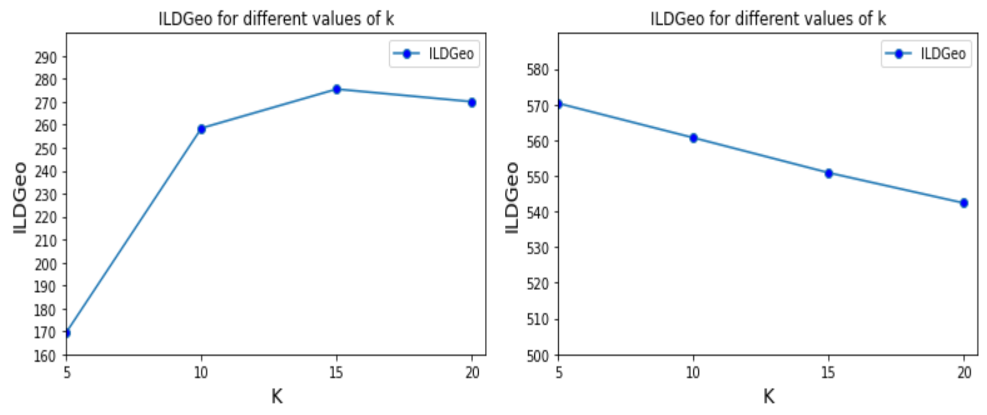


Figure 10. Comparison of ILDGeo between Tallinn and New York dataset (left to right)

for each user. As we can see from figure9, the coverage value is increasing as the value of k increases. It can be explained by the increase in the item number in the recommendation list with the value of k. In addition, figure9 (left for Tallinn) has higher coverage than figure9 (right for New York), because the number of items/POIs is much higher in the New York dataset.

The ILDGeo metric reflects the average kilometer distance between all the recommended items. Thus, when the distance is low, the system is more likely to recommend items that are closer to the users.

From figure 10 (left for Tallinn), the ILDGeo value for $k = 5$ is 169.47, being the lowest. So, the recommended locations are close to the users. When the value of k gets increased $k = 10$, $k = 15$, $k = 20$, the value of ILDGeo also got increased as well 258.53, 275.55, and 270.07 respectively. However, for figure 10 (right for New York), our approach has the highest value of ILDGeo with a value of 570.29 for $k = 5$. With the increase of k to 10, 15, 20, the value of ILDGeo decreases to 560.64, 550.84, and 542.35 respectively.

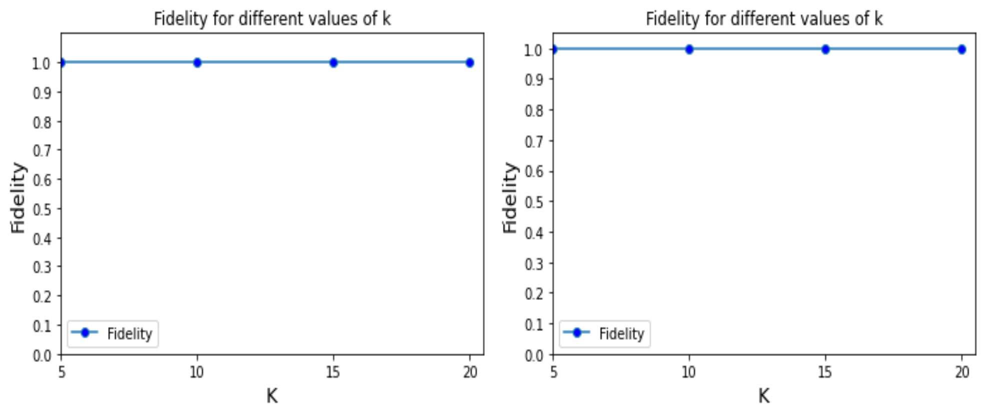


Figure 11. Comparison of Fidelity between Tallinn and New York dataset (left to right)

The behavior of ILDGeo values is different for Tallinn and New York datasets since our system is recommending items based on relevance and unexpectedness not based on the geographical location of each item.

The fidelity measure shows how many explainable/relevant items we have in our recommended list. Explainable items are those which are extracted from users based on the commonality of the visited locations (called relevance in our Relevance algorithm 1). Since our recommendation list includes only relevant items, the fidelity value of our approach is always equal to 1 no matter the dataset, as shown in figure11.

Performance Evaluation against Competitor Models

In the following, we perform a comparison of our model with other baseline models for Tallinn and New York datasets.
As we see from figure 12 (left for Tallinn), our system (Serendipity model) is consistently performing well in precision as compared to competing models, for $k = 5$, 10, 15, and 20 with the respective values equal to 0.1121, 0.0847, 0.0715, and 0.0607. It means that our system is providing more precise recommendation lists as compared to others. Competing models' values for $k = 5$, 10, 15, and 20 are as follows. First, for the Rank-GeoFM model the values were 0.005423, 0.005552, 0.005552, and 0.005326 respectively. For the LGLMF model, we obtained the respective values 0.0612, 0.0423, 0.0333, and 0.02788. Finally, the STACP model's results were equal to 0.04222, 0.0320, 0.0255, and 0.0214 respectively.
In addition, for the New York dataset, c.f. figure 12 (right for New York), our system is also performing consistently well as compared to competing models for different values of $k = 5$, 10, 15, 20 with the respective values equal 0.0481, 0.0383, 0.0305, and 0.0278. However, competing model's values for $k = 5$, 10, 15, 20 are as follows. For example,

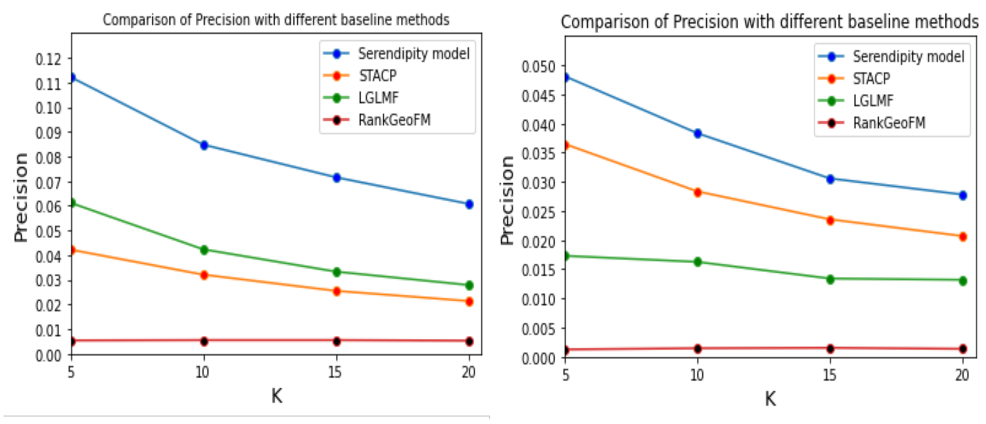


Figure 12. Comparison of Precision between Tallinn and New York dataset (left to right)

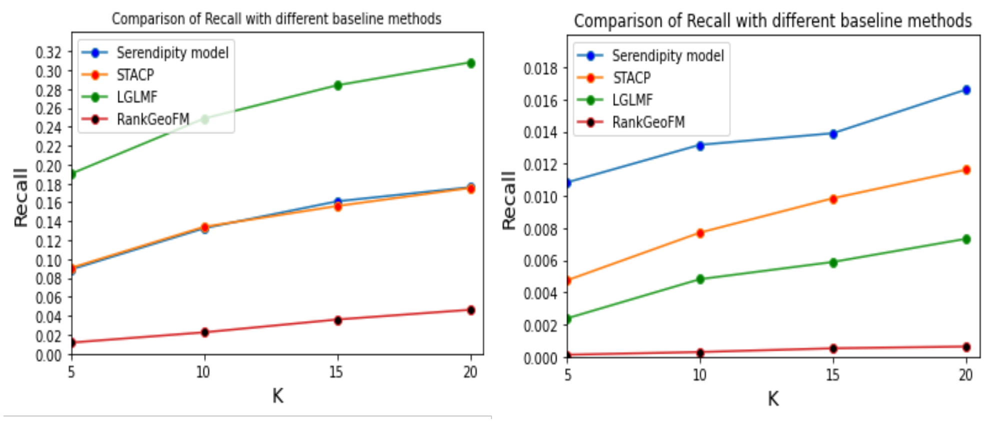


Figure 13. Comparison of recall between Tallinn and New York dataset (left to right)

the Rank-GeoFM approach achieved the respective results equal to 0.001251, 0.001460, 0.001529, and 0.001356. In terms of recall value, figure 13, our system is consistently improving for graph (left for Tallinn) and gives almost the same recall value as of the STACP model. Values of recall are 0.088, 0.1325, 0.1610, and 0.1759 for $k = 5$, 10, 15, 20 respectively. However, competing models' values for $k = 5$, 10, 15, 20 are lower. For example, for the Rank-GeoFM model, the values were 0.011636, 0.022496, 0.036070, and 0.046361, respectively. However, for the LGLMF model, we got the respective values 0.19005, 0.24855, 0.2837, and 0.3079. Finally, the STACP model's results were equal to 0.0905, 0.13404, 0.156087, and 0.175198 respectively.

In addition, for New York dataset (c.f. figure 13 (right for New York), our system is outperforming all the competing models and continuously improving as well. Values of recall are 0.0108, 0.0131, 0.0138, and 0.0166 for $k = 5$, 10, 15, 20 respectively. However, competing models' values for $k = 5$, 10, 15, 20 are lower than the one got with our approach. For example, the Rank-GeoFM achieved, respectively, results equal to 0.000114, 0.000284, 0.000515, and 0.000625 .

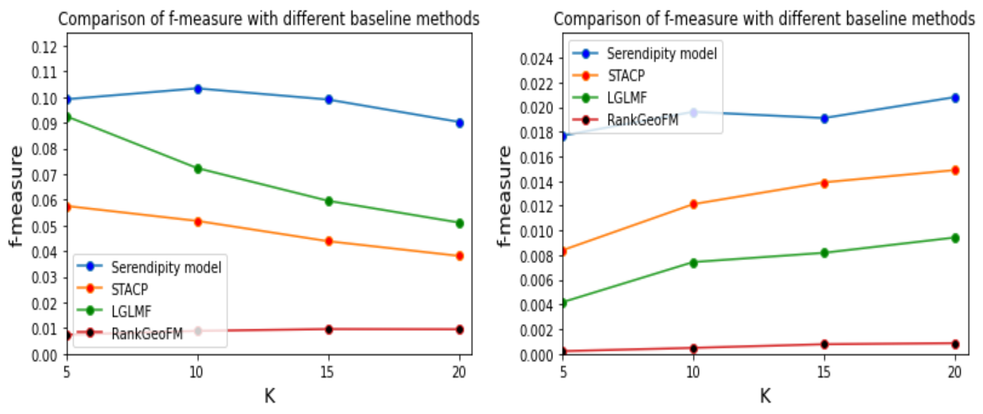
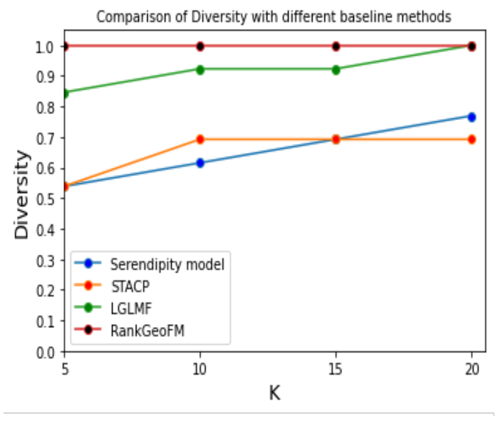


Figure 14. Comparison of f-measure between Tallinn and New York dataset (left to right)

As described before, our goal is not to look only for precision or recall values but is to balance both metrics and the f-measure gives us that balance. As depicted in figure 14 (left for Tallinn), our model is performing very well as compared to the competing models. With the increase of k value, our system gives us the best f-measure values. Values for $k = 5$, 10, 15, 20 are 0.9917, 0.1034, 0.0990, and 0.0903 respectively. However, competing models' values for $k = 5$, 10, 15, 20 are as follows. For example, the Rank-GeoFM model gives the following respective values 0.0073, 0.0089, 0.0096, and 0.0095. Also, for the LGLMF model, we obtained the values equal to 0.0925, 0.0723, 0.0596, and 0.0511 respectively.
For figure 14 (right for New York), our model is also performing consistently well as compared to the competing models. For $k = 5$, 10, 15, 20 values are 0.0176, 0.01960, 0.0191, and 0.0207 respectively. With the increase of k, the f-measure values are also increasing. Thence, our system has a better balance between precision and recall. However, the competing models' values for $k = 5$, 10, 15, 20 were worse. For instance, The LGLMF model f-measure values were equal to 0.00417, 0.00742, 0.00818, and 0.00942 respectively.

As for diversity, that shows how diverse recommendation items are in terms of their categories, figure 15 (left for Tallinn) proves that our system is continuously improving in terms of diversity for different values of k. For values of $k = 5$, 10, 15, and 20, our approach achieve good diversity results equal to 0.5384, 0.6153, 0.6923, and 0.7692 respectively. However, the best diversity results were given by the Rank-GeoFM model which achieved a 100% diversity for $k = 5$, 10, 15, and 20. Furthermore, as sketched in figure 15 (right for New York), the diversity of our model is continuously increasing as well. For values of $k = 5$, 10, 15, 20 values are 0.272, 0.38, 0.488, and 0.572 respectively. However, for the New York dataset, the competing models' achieved better results for this evaluation metric. The best results were obtained with Rank-GeoFM model as well with the diversity values equal to 0.872, 0.928, 0.944, and 0.952 respectively.



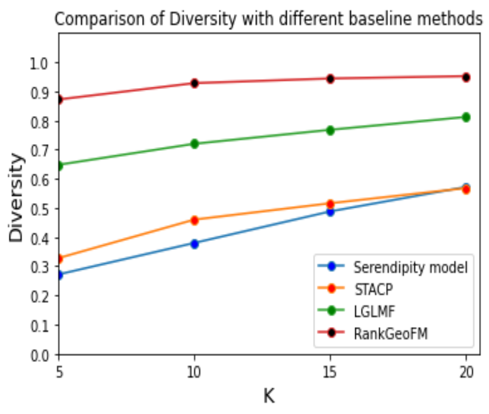


Figure 15. Comparison of diversity between Tallinn and New York dataset (left to right)

Since our system recommends items that are relevant and unexpected at the same time without considering their diversity. Items may belong to the same category as other items. By looking at figure 15, our system is continuously improving in terms of diversity.

For the comparison of the coverage metric results with the competing models, we can see from figure 16 that the coverage for our model is increasing by incresing k. As the coverage represents how many POIs our systems have covered among all the POIs. As for figure 16 (left for Tallinn) $k = 5$, 10, 15, and 20, our system coverage values are equal to 0.1212, 0.24009, 0.3403, and 0.4009 respectively.

And for figure 16 (right for New York) our model has lowest coverage among competing models. For values of $k = 5$,10 ,15, and 20, we obtained only 0.00519, 0.0106, 0.0174, and 0.02431 respectively. However, the Rank-GeoFM model coverage achieved 0.10964, 0.20134, 0.280463, and 0.34922 respectively.

As described before, our system recommends items which are relevant and unexpected at the same time, it can recommend the same items to multiple users, hence accounting for lower overall coverage.

As depicted in figure 17 (left for Tallinn), our system has the lowest ILDGeo values for different k values. It means that our system is recommending items closer to the user as compared to other models' ILDGeo values. The lowest the ILGDeo values, the closer the recommended items are to the user. For values of $k = 5$, 10, 15, 20 ILGDeo values are 169.47, 258.53, 275.55, and 270.07 respectively.

Moreover, as shown in figure 17 (right for New York), our system has the lowest ILDGeo values compared to other models for New York dataset too. For example, the highest ILDGeo values were obtained with the Rank-GeoFM model with respective values equal to 6304.78, 6408.42, 6375.34, and 6402.80 respectively for $k = 5$, 10, 15, 20.

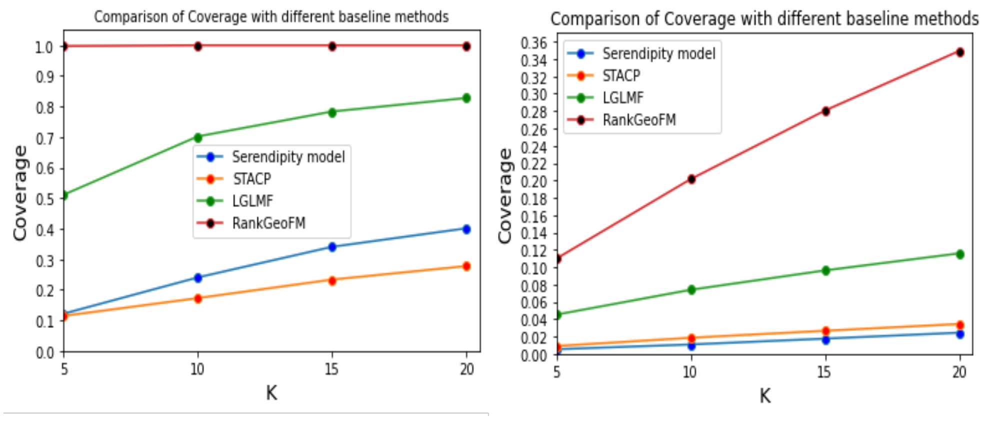


Figure 16. Comparison of coverage between Tallinn and New York dataset (left to right)

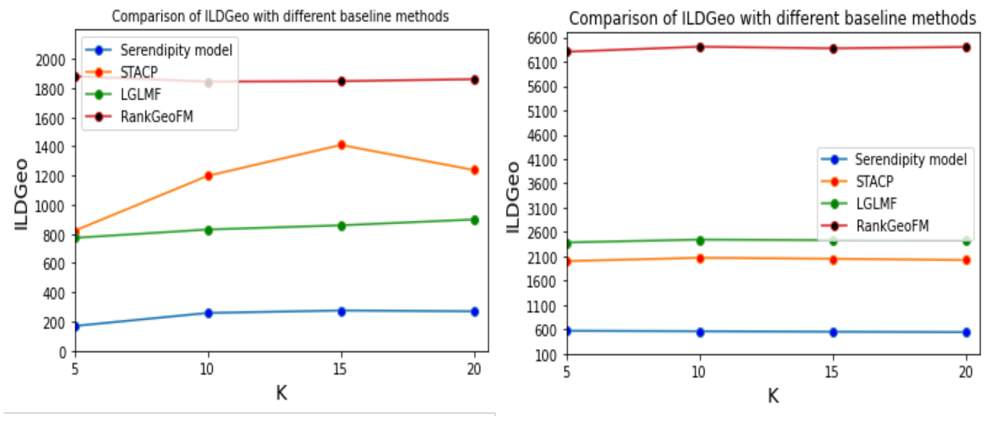


Figure 17. Comparison of ILDGeo between Tallinn and New York dataset (left to right)

As shown in figure 18 (left and right), our model has a consistent max fidelity value of 1.0 for both datasets. The reason is that since fidelity tells us the number of relevant items in a recommended list and sine our system recommends only relevant/unexpected items, the resulting recommended list is always relevant and unexpected to a user as well. Hence, accounted for max fidelity.

While observing competing models' fidelity values, we notice that their values are decreasing.

For example, as seen in figure 18 (left for Tallinn and right for New York), the competing models' values for $k = 5$, 10, 15, and 20. For instance, for New York dataset, the STACP model fidelity decreased from 0.32781 for $k = 5$ to reach 0.19699 for $k = 20$.

Evaluation of Serendipitous Recommendation based on Relevance, Unexpectedness, and Novelty

In this section, we study the impact of incorporating the novelty metric into the serendipity definition based on the Tallinn dataset. We use the metric of novelty along with relevance

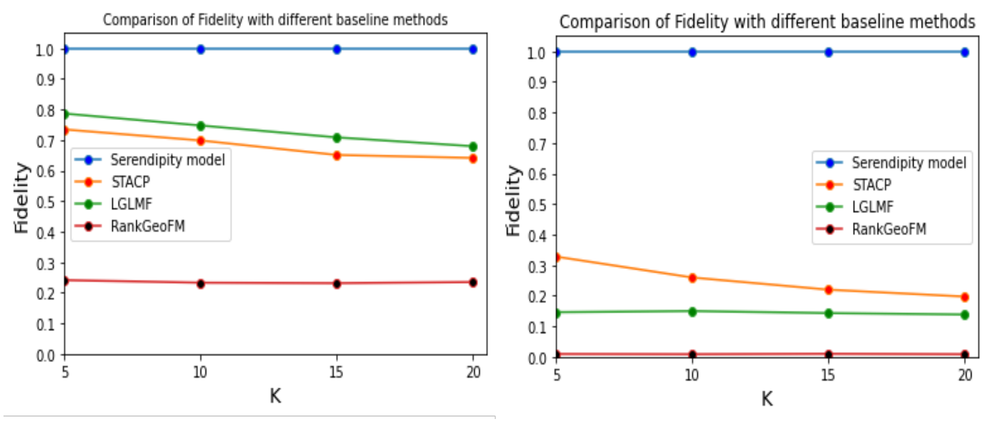


Figure 18. Comparison of Fidelity between Tallinn and New York dataset (left to right)

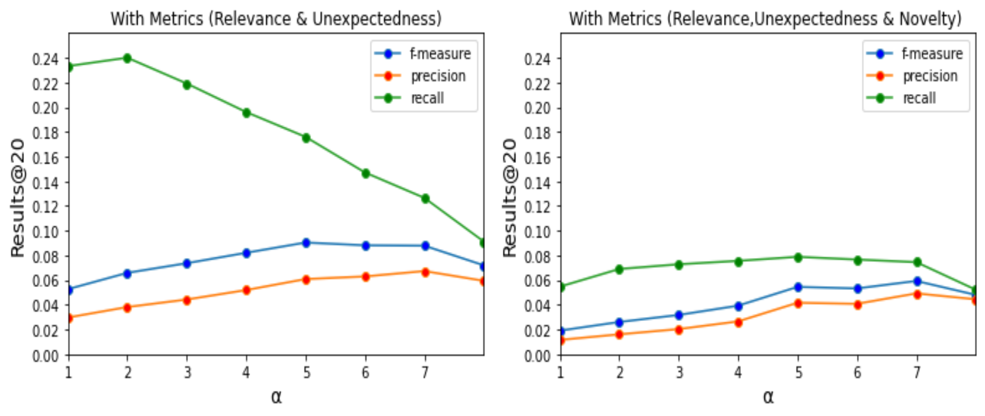


Figure 19. Performance of our model by including the metrics (Relevance and Unexpectedness) and the metrics (Relevance, Unexpectedness, and novelty) for different values of parameter α

and unexpectedness to generate recommended items considered serendipitous since they are relevant, unexpected, and novel at the same time.

Figure 19 shows the performance comparison for two variants of our approach using the metrics (Relevance, unexpectedness) and the metrics (Relevance, unexpected, and novelty) for $k = 20$. As seen from figure 19 (left), when the value of $\alpha = 1$, the f-measure, precision, and recall are equal to 0.0525, 0.0296, and 0.2331 respectively, as compared to figure 19 (right) when the f-measure, precision, and recall are equal respectively to 0.0190, 0.0115, and 0.0546. We got the lowest values by including the novelty metric to the serendipity computation. For example, when we set $\alpha = 2$, the values are lower for graph (right) as compared to graph(left). The optimal parameter for figure 19 (left) is $\alpha = 5$, with an f-measure, a precision, and a recall of 0.0903, 0.0607, and 0.17595 respectively, as compared to figure 19 (right) with the respective values 0.05446, 0.0416, and 0.0788. The same for $\alpha = 5$, figure 19 (left) shows better results with an f-measure, a precision, and a recall of 0.0878, 0.0672, 0.126 respectively, as compared to figure 19 (right) with the respective values equal to 0.059, 0.049, and 0.0745.

To sum up, the novelty impact is negative for generating serendipitous POIs based on the Tallinn dataset.

4.4 Conclusion

In this chapter, we discussed in the detail our algorithms for relevance, unexpectedness, and novelty. By using the relevance and unexpectedness of the metric on two datasets, we notice that our model is behaving better than compared to other models in terms of f-measure, precision, recall, fidelity, ILDGeo and also observed the impact of other metrics as well. Later, we include the metric novelty and observed that it has a negative impact on the results.

5. Conclusion and Future Work

In this thesis, we suggested a novel Serendipity-based POI RS model which aims to recommend user relevant and unexpected POIs. This model uses relevance metric to recommend relevant POIs to each user based on his/her check-in history and among that recommended relevant list, it computes the unexpectedness of each POI using the metric Unexpectedness. Experimental results using two different datasets, mainly Flickr Tallinn and New York Foursquare, show that our model is performing well in terms of accuracy and beyond accuracy evaluation. In addition, we studied the impact of incorporating the novelty metric in the definition of the serendipity along with relevance and unexpectedness. The obtained results based on the Tallinn dataset showed that the novelty does not have a good impact on our model.

As future work, we plan to apply our approach on large scale datasets and enhance the efficiency of the algorithm, which can handle large datasets with ease. For new users and items, our future goal is to provide recommendations for that as well by solving the cold start problem. At last, our goal is to incorporate more user related information into our algorithm which can be extracted from online social networks.

Bibliography

[1] Imen Ben Sassi, Sehl Mellouli, and Sadok Ben Yahia. "Context-aware recommender systems in mobile environment: On the road of future research". In: *Inf. Syst.* 72 (2017), pp. 27–61. DOI: 10.1016/j.is.2017.09.001. URL: https://doi.org/10.1016/j.is.2017.09.001.

[2] J. Ben Schafer, Joseph Konstan, and John Riedl. "Recommender Systems in E-Commerce". In: *Proceedings of the 1st ACM Conference on Electronic Commerce*. EC '99. Denver, Colorado, USA: Association for Computing Machinery, 1999, pp. 158–166. ISBN: 1581131763. DOI: 10.1145/336992.337035. URL: https://doi.org/10.1145/336992.337035.

[3] Jesus Bobadilla et al. "Recommender systems survey". In: *Knowledge-Based Systems* 46 (July 2013), pp. 109–132. DOI: 10.1016/j.knosys.2013.03.012.

[4] Oscar Celma. *Music Recommendation and Discovery - The Long Tail, Long Fail, and Long Play in the Digital Music Space.* Jan. 2010. ISBN: 978-3-642-13286-5. DOI: 10.1007/978-3-642-13287-2.

[5] Daniel Lewis. "What is Web 2.0?" In: *XRDS* 13.1 (Sept. 2006), p. 3. ISSN: 1528-4972. DOI: 10.1145/1217666.1217669. URL: https://doi.org/10.1145/1217666.1217669.

[6] Shuguang Cui et al. *Big Data over Networks*. Cambridge University Press, 2016.

[7] *Flickr, Online: https://en.wikipedia.org/wiki/Flickr [Accessed 07-05-2021.*

[8] Jean B. McGrew. "Future Shock. Alvin Toffler. New York: Random House, 1970. 505 pp. $7.95". In: *The bulletin of the National Association of Secondary School Principals* 54.349 (1970), pp. 123–129. DOI: 10.1177/019263657005434912. eprint: https://doi.org/10.1177/019263657005434912. URL: https://doi.org/10.1177/019263657005434912.

[9] Bin Liu and Hui Xiong. "Point-of-Interest Recommendation in Location Based Social Networks with Topic and Location Awareness". In: May 2013, pp. 396–404. ISBN: 978-1-61197-262-7. DOI: 10.1137/1.9781611972832.44.

[10] Imen Ben Sassi and Sadok Ben Yahia. “How does context influence music preferences: a user-based study of the effects of contextual information on users’ preferred music”. In: *Multim. Syst.* 27.2 (2021), pp. 143–160. DOI: 10.1007/s00530-020-00717-x. URL: https://doi.org/10.1007/s00530-020-00717-x.

[11] Denis Kotkov, Jari Veijalainen, and Shuaiqiang Wang. “Challenges of serendipity in recommender systems”. In: *WEBIST 2016: Proceedings of the 12th International conference on web information systems and technologies. Volume 2, ISBN 978-989-758-186-1*. SCITEPRESS. 2016.

[12] Eugenio Tacchini. “Serendipitous mentorship in music recommender systems”. In: (2012).

[13] Qian Meng and Kenji Hatano. “Visualizing Basic Words Chosen by Latent Dirichlet Allocation for Serendipitous Recommendation”. In: *Proceedings - 2014 IIAI 3rd International Conference on Advanced Applied Informatics, IIAI-AAI 2014* (Sept. 2014), pp. 819–824. DOI: 10.1109/IIAI-AAI.2014.164.

[14] Mehrbakhsh Nilashi et al. “Collaborative Filtering Recommender Systems”. In: *Research Journal of Applied Sciences, Engineering and Technology* 5 (Apr. 2013), pp. 4168–4182. DOI: 10.19026/rjaset.5.4644.

[15] G. Adomavicius and A. Tuzhilin. “Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions”. In: *IEEE Transactions on Knowledge and Data Engineering* 17.6 (2005), pp. 734–749. DOI: 10.1109/TKDE.2005.99.

[16] Francesco Ricci, Lior Rokach, and Bracha Shapira. “Recommender systems: introduction and challenges”. In: *Recommender systems handbook*. Springer, 2015, pp. 1–34.

[17] Peter Brusilovsky and Mark Maybury. “The adaptive Web”. In: *Communications of the ACM* 45 (Jan. 2002), pp. 30–33.

[18] Francesco Ricci, Lior Rokach, and Bracha Shapira. “Introduction to recommender systems handbook”. In: *Recommender systems handbook*. Springer, 2011, pp. 1–35.

[19] J. Ben Schafer et al. “Collaborative Filtering Recommender Systems”. In: *The Adaptive Web: Methods and Strategies of Web Personalization*. Ed. by Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 291–324. ISBN: 978-3-540-72079-9. DOI: 10.1007/978-3-540-72079-9_9. URL: https://doi.org/10.1007/978-3-540-72079-9_9.

[20] Shanshan Feng et al. “Personalized ranking metric embedding for next new poi recommendation”. In: *IJCAI’15 Proceedings of the 24th International Conference on Artificial Intelligence*. ACM. 2015, pp. 2069–2075.

[21] Chenyi Zhang et al. “Personalized trip recommendation with poi availability and uncertain traveling time”. In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 2015, pp. 911–920.

[22] Claudio Lucchese et al. “How random walks can help tourism”. In: *European Conference on Information Retrieval*. Springer. 2012, pp. 195–206.

[23] Quan Yuan et al. “Time-aware point-of-interest recommendation”. In: *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 2013, pp. 363–372.

[24] Leo Iaquinta et al. “Introducing serendipity in a content-based recommender system”. In: *2008 Eighth International Conference on Hybrid Intelligent Systems*. IEEE. 2008, pp. 168–173.

[25] Allen Foster and Nigel Ford. “Serendipity and information seeking: an empirical study”. In: *Journal of documentation* (2003).

[26] Paul André et al. “Discovery is never by chance: designing for (un) serendipity”. In: *Proceedings of the seventh ACM conference on Creativity and cognition*. 2009, pp. 305–314.

[27] *Serendipity - definition of serendipity by the free dictionary, Online: https://www.thefreedictionary.com/ [Accessed 08-05-2021*.

[28] Yuan Cao Zhang et al. “Auralist: introducing serendipity into music recommendation”. In: *Proceedings of the fifth ACM international conference on Web search and data mining*. 2012, pp. 13–22.

[29] Andrii Maksai, Florent Garcin, and Boi Faltings. “Predicting online performance of news recommender systems through richer evaluation metrics”. In: *Proceedings of the 9th ACM Conference on Recommender Systems*. 2015, pp. 179–186.

[30] Sean M. McNee, John Riedl, and Joseph A. Konstan. “Being Accurate is Not Enough: How Accuracy Metrics Have Hurt Recommender Systems”. In: *CHI ’06 Extended Abstracts on Human Factors in Computing Systems*. CHI EA ’06. Montréal, Québec, Canada: Association for Computing Machinery, 2006, pp. 1097–1101. ISBN: 1595932984. DOI: 10.1145/1125451.1125659. URL: https://doi.org/10.1145/1125451.1125659.

[31] Jonathan L Herlocker et al. “Evaluating collaborative filtering recommender systems”. In: *ACM Transactions on Information Systems (TOIS)* 22.1 (2004), pp. 5–53.

[32] Adamopoulos Panagiotis and T Alexander. “On unexpectedness in recommender systems: Or how to expect the unexpected”. In: *Workshop on Novelty and Diversity in Recommender Systems, 2011*. 2011.

[33] Chen Ma et al. "Point-of-interest recommendation: Exploiting self-attentive autoencoders with neighbor-aware influence". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2018, pp. 697–706.

[34] Fei Sun et al. "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer". In: *Proceedings of the 28th ACM international conference on information and knowledge management*. 2019, pp. 1441–1450.

[35] Q. Liu et al. "Predicting the Next Location: A Recurrent Model with Spatial and Temporal Contexts". In: *AAAI*. 2016.

[36] Maurizio Ferrari Dacrema et al. "Critically examining the claimed value of convolutions over user-item embedding maps for recommender systems". In: *arXiv preprint arXiv:2007.11893* (2020).

[37] Maritzol Tenemaza et al. "Improving itinerary recommendations for tourists through metaheuristic algorithms: an optimization proposal". In: *IEEE Access* 8 (2020), pp. 79003–79023.

[38] Xia Ning, Christian Desrosiers, and George Karypis. "A comprehensive survey of neighborhood-based recommendation methods". In: *Recommender systems handbook* (2015), pp. 37–76.

[39] Marco De Gemmis et al. "An investigation on the serendipity problem in recommender systems". In: *Information Processing & Management* 51.5 (2015), pp. 695–717.

[40] L Lovász. "Random walks on graphs: a survey, in "Combinatorics, Paul Erds is eighty", Vol. 2, 353–397, János Bolyai Math". In: *Soc., Budapest* (1996).

[41] Marco Gori et al. "Itemrank: A random-walk based scoring algorithm for recommender engines." In: *IJCAI*. Vol. 7. 2007, pp. 2766–2771.

[42] Hilmi Yildirim and Mukkai S Krishnamoorthy. "A random walk method for alleviating the sparsity problem in collaborative filtering". In: *Proceedings of the 2008 ACM conference on Recommender systems*. 2008, pp. 131–138.

[43] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. "Content-based recommender systems: State of the art and trends". In: *Recommender systems handbook* (2011), pp. 73–105.

[44] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

[45] Tomoko Murakami, Koichiro Mori, and Ryohei Orihara. "Metrics for evaluating the serendipity of recommendation lists". In: *Annual conference of the Japanese society for artificial intelligence*. Springer. 2007, pp. 40–46.

[46] Hossein A Rahmani et al. “Joint geographical and temporal modeling based on matrix factorization for point-of-interest recommendation”. In: *European Conference on Information Retrieval*. Springer. 2020, pp. 205–219.

[47] Arthur W Hafner. “Pareto’s principle: The 80-20 rule”. In: *Retrieved December* 26 (2001), p. 2001.

[48] Huiji Gao et al. “Exploring temporal effects for location recommendation on location-based social networks”. In: *Proceedings of the 7th ACM conference on Recommender systems*. 2013, pp. 93–100.

[49] Xutao Li et al. “Rank-geofm: A ranking based geographical factorization method for point of interest recommendation”. In: *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 2015, pp. 433–442.

[50] Jason Weston, Samy Bengio, and Nicolas Usunier. “Large scale image annotation: learning to rank with joint word-image embeddings”. In: *Machine learning* 81.1 (2010), pp. 21–35.

[51] Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. “A survey of serendipity in recommender systems”. In: *Knowledge-Based Systems* 111 (2016), pp. 180–192.

[52] Saúl Vargas and Pablo Castells. “Rank and relevance in novelty and diversity metrics for recommender systems”. In: *Proceedings of the fifth ACM conference on Recommender systems*. 2011, pp. 109–116.

[53] Jean-Benoıt Griesner. “Scalable models for Points-Of-Interest recommender systems”. PhD thesis. Telecom ParisTech, 2018.

[54] Marius Kaminskas and Derek Bridge. “Diversity, serendipity, novelty, and coverage: a survey and empirical analysis of beyond-accuracy objectives in recommender systems”. In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 7.1 (2016), pp. 1–42.

[55] Huida Jiao, Fan Mo, and Hayato Yamana. “Evaluation of POI recommendation system beyond accuracy: Diversity explainability and computation cost”. In: *Proceedings of 18th Japan data engineering and information management (DEIM) forum*. 2020.

[56] Behnoush Abdollahi and Olfa Nasraoui. “Using explainability for constrained matrix factorization”. In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 2017, pp. 79–83.

[57] *Flickr, Online: https://deepai.org/machine-learning-glossary-and-terms/f-score [Accessed 01-05-2021.*

Appendices

Appendix 1 - Non-exclusive licence for reproduction and publication of a graduation thesis [1]

[1]The non-exclusive licence is not valid during the validity of access restriction indicated in the student’s application for restriction on access to the graduation thesis that has been signed by the school’s dean, except in case of the university’s right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.