VRIJE
UNIVERSITEIT
BRUSSEL

Graduation thesis submitted in partial fulfilment of the requirements for the degree of Master in Computer Sciences and Engineering

HEURISTIC REAL-TIME TRAJECTORY PREDICTION USING SPATIO-TEMPORAL DATABASES

Ahmed Barhdadi

September 2023

Supervisor: Prof. Dr. Beat Signer
Advisor: Maxim Van de Wynckel

Sciences and Bio-Engineering Sciences

Abstract

This thesis focuses on trajectory prediction utilising historical data and contextual information in indoor environments. The main objective of this thesis is to create a heuristic trajectory prediction system using the MobilityDB, which is a spatio-temporal trajectory database that excels at handling trajectory data. The basis of the prediction system is MobilityDB's ability to efficiently store and query trajectory data while considering both the spatial and temporal dimensions. This approach allows for transparent and accurate predictions in indoor environments. By using graph link prediction and taking temporal habits into account, we improve the prediction accuracy even more, and we can provide a thorough understanding of users' movement patterns and habits.

We have thoroughly tested the effect of the *temporal habit factor* and the addition of new habits to the dataset to see how well the trajectory prediction system would perform. The results we obtained show that taking the temporal dimension and temporal habits into account improves the model's prediction accuracy. However, due to the limitation of the physical space of the indoor environment, some predictions were inevitably predicted inaccurately. Nonetheless, the model still performed reasonably well, even when new (different) habits were added to the database next to the already existing ones. The evaluation suggests that the model is not entirely able to adapt quickly to changing habits. However, even when the database expands, it remains flexible and maintains a decent prediction accuracy.

Acknowledgements

Firstly, I want to thank the promotor of this topic who helped review this thesis, Beat Signer. A special thanks is given to the advisor Maxim Van de Wynckel for his continuous help and feedback throughout the year. I would also like to express my gratitude to Bram Vandenbogaerde, who provided invaluable assistance during particularly challenging moments. Lastly, I am deeply thankful to my family and friends for their unwavering support and endless encouragement.

1 Introduction

Because of its wide applications in fields such as transportation, security, and social networks, trajectory prediction has become an important research topic in recent years. In this thesis, we focus on the subject of trajectory prediction in indoor and outdoor environments using historical data and contextual information.

Our approach is based on an intentional choice to not use traditional machine learning approaches. Although machine learning has great capabilities, it needs a large amount of training data and can occasionally operate as a "black box," making it impossible to understand the reasoning behind its predictions. Our problem is reliably predicting a trajectory in various surroundings without relying on complex approaches that demand a lot of data, are challenging to comprehend, and are unreliable. How can we predict trajectories with transparency and accuracy?

Our heuristic methodology, on the other hand, is based on clear and traceable patterns, providing transparency and flexibility. It allows for exact predictions based on observed behaviour without the need for large datasets.

Our system utilises MobilityDB, a specialised database designed for trajectory data, to accurately predict recurring trajectories (habits). By incorporating both spatial and temporal dimensions, we can increase the precision of our predictions, ensuring accuracy in various real-world scenarios. If someone habitually visits a certain place at a specific time, our system will recognise and use this habit to anticipate their typical behaviour when they are near this location at that time.

One of the novel contributions of our study is the use of MobilityDB as a spatio-temporal database for trajectory prediction, which has not been researched before. MobilityDB enables us to store and retrieve trajectory data with geospatial and temporal dimensions, which is required for accurate trajectory prediction in real-world scenarios.

In addition to these contributions, we aim to:

- research the various possibilities MobilityDB might bring for trajectory prediction.
- perform a thorough examination of how our heuristic prediction compares to other commonly used methods.
- evaluate our system's robustness in a range of conditions, including the inherent errors or "noise" found in real-world data.
- present a novel technique for trajectory prediction that combines graph link prediction and takes temporal habits into account, providing a more complete picture of movement patterns.

Contents

2 Background

Knowing how an object or person will move in the future based on their prior movements is known as trajectory prediction. The basic goal is to draw conclusions from historical data and identify patterns in either human or object behaviour. These patterns make it possible to make precise predictions about how these objects will move in the future.

2.1 Applications

The amount of the mobility data opens up new possibilities for identifying human motion patterns and performing predictions on future trajectories [1]. To be able to understand and predict human mobility is considered to be crucial for various applications such as urban planning, traffic management and updates, suggested routes, location-based advertisement, collision avoidance of air crafts and maritime ships and many more [1, 2, 3]. More concrete, Chen et al. [4] mention an example where it is highly desirable to have a high accuracy regarding the prediction that is made. It would be possible to estimate traffic conditions and give drivers more logical paths to avoid or relieve traffic bottlenecks by anticipating where vehicles will be on the road next.

Although some researchers predict connectivity to GSM mobile phone towers, some concentrate on predictions using Wi-Fi networks. In road networks, where movements are restricted to roads on the map, trajectory prediction is utilised in addition to the mobility patterns of smartphone users. In smart home environments, location prediction technologies are also employed to enhance comfort and reduce unnecessary expenses [5].

2.2 Challenges

Depending on the application where trajectory prediction is used, the challenges will vary. Real-time prediction is very important because autonomous vehicles must act quickly to prevent collisions and guarantee passenger safety. Dealing with uncertainty, which can result in faulty trajectory prediction due to inaccurate sensor data, is another difficulty. The difficulty of managing complicated traffic situations, like intersections, where several vehicles must work together to avoid collisions, is yet another important challenge [6, 7]. A study of trajectory-prediction techniques for autonomous driving is provided by Huang et al. [6], who also noted the various difficulties of trajectory prediction in this particular scenario. The authors classify popular and known trajectory prediction methods in four different classes after looking into the various existing prediction methods over the past two decades, which can be seen in Figure 1. All the methods, except for the physics-based approach, are mainly machine learning methods.

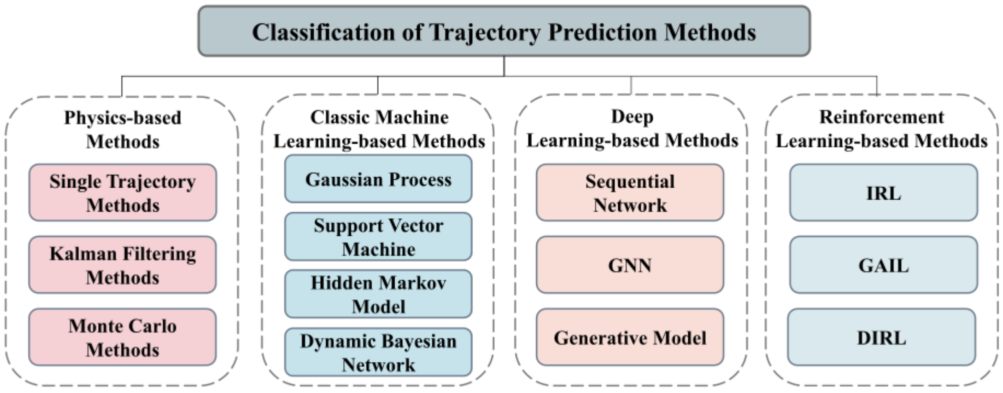

Figure 1: Trajectory prediction categories [6]

The use of machine learning techniques and models is by no means the only option for trajectory prediction approaches. Existing models such as the physics-based methods on Figure 1, Markov chains, historical database querying, using data mining to find patterns (when a lot of data is available). In Section 3, we will dive deeper into both the ML and non-ML methods and techniques.

2.2.1 Uncertainty

Because many real-world settings are unknown, trajectory prediction is a difficult task. Many factors, including noisy sensor data, unpredictable human behaviour, and unforeseen environmental events, might contribute to this uncertainty. The topic of uncertainty in trajectory prediction is mentioned in several of the studies.

In the paper by Huang et al. [6] regarding autonomous vehicles, the authors point out that in autonomous driving scenarios, it is challenging to precisely predict the trajectories of nearby cars due to the presence of uncertainties. They suggest a technique that explicitly models uncertainty and includes it into the trajectory-prediction technique using a Bayesian framework.

Similar to the previous example, Sadri et al. [5] emphasise the significance of taking uncertainty into consideration when predicting the trajectory of pedestrians. To be able to predict future movements and also take uncertainty into account, they suggest a technique that makes use of a Markov model.

Chen et al. [4] cover the same issue by suggesting the use of a recurrent neural network, a machine-learning model that learns from previous trajectories. They then make an effort to produce probabilistic predictions of future trajectories.

The above-mentioned papers all highlight how essential it is to take uncertainty into account when predicting future trajectories, and how crucial it is to create techniques that can deal with this uncertainty. In order to produce predictions with a high accuracy that take this problem in real-world contexts into account, effective trajectory-prediction approaches will need to be used to deal with this.

2.3 Spatio-temporal Databases

In this section, we will shortly discuss spatial databases, temporal databases and the combination of the two, namely spatio-temporal databases.

2.3.1 Spatial Databases

A spatial database is a database that is designed to store and query data about objects in space, such as points, lines, and polygons. While most databases understand various numeric and character data types, additional features are necessary to understand these geographical data types [8].

While traditional DBMS (Database Management Systems) can manage large datasets and give resilience against failures, they are inefficient for dealing with geographical data and queries. Shukla et al. provide an example where a non-spatial query would return a list of bookshops based on the book inventory, but a spatial query might return bookshops within a given distance. A spatial database is required here to be able to understand and manage geographical data types [9]. This is a straightforward example, but in today's world, the importance goes far beyond the example with a diverse range of applications ranging from location-based services to Geographic Information Systems (GIS) and beyond.

2.3.2 Temporal Databases

Conventional DBMS are not suitable for storing and processing data that changes over time in the real world as handling data values from the past, present and future that contain historical information is not evident. Jaymin Patel mentions three different "types" of data, namely *current* data, *past* data and *future* data. Current data is referred to as data stored by the system that is valid at the present time. Past data contains information that was stored at an earlier moment, which used to be valid around that time. Future data is defined as data that will eventually become true. Databases like Oracle and Postgres do not support query operations to handle past and future data [10].

However, temporal databases are different in the fact that they are designed to handle time-varying data. Allowing data storage with timestamps makes it possible to understand how the data changes over time and query across various time periods. This is a necessity for applications that need the tracking of historical changes, the prediction of future trends, the analysis of time series data, location-based services and trajectory prediction, where understanding past patterns aids in predicting future movements [11, 12].

Some of the unique features and characteristics of temporal databases include storage and management, time-based querying of temporal data and efficient handling of time series data.

Temporal databases allow users to query data based on time, which makes it possible to retrieve information about the state of the database at a single point in time or across a period of time. This functionality is especially useful for applications that require the analysis of changes over time, such as financial analyses, traffic management, urban planning, trajectory prediction and more [13].

A time series query is able to detect sequences of data points that occur at distinct increasing times. These criteria can range from statistical rules to temporal patterns to time series similarities. Particularly, similarity queries play an important role in real-world applications such as stock research and weather forecasting, which deal with enormous amounts of time series data [14].

2.4 Spatio-temporal Databases

After having discussed the various features and characteristics of spatial and temporal databases, we now have a look at spatio-temporal databases, which combines the two. It is critical to consider the unique properties of space and time when creating an efficient representation of them. However, there are several viewpoints on how to see space and time. This is the key problem in building a unified method that can serve a variety of applications [15].

Spatial data is any data item that is associated to space (location-aware or geotagged). Historically, recognised patterns of spatial data included raster data (e.g. satellite pictures), point data (e.g. crime reports), and network data (e.g. road maps). However, with the widespread availability of GPS-enabled devices and location-based services, the nature of spatial data has changed. Check-ins, geotagged tweets, and GPS trajectories have now been added to the mix. One obtains spatio-temporal data when combining spatial information with timestamps. It can be divided into two types: discrete point data, which pinpoints specific occurrences at a certain time and place (such as the position and timing of a traffic collision), and trajectory data, which monitors movement over time, such as a taxi's route throughout the day [12].

Where does the need for spatio-temporal databases come from? There exist many real-world applications, in which this is a necessity. There are many applications which show us why spatio-temporal databases are essential. We now show a couple of examples provided by Alam et al. that show the importance and utility [12]:

Applications and Utility of Spatio-temporal Databases:

- **Epidemiology and Public Health**:
  - Track and analyse disease spread patterns.
  - Enhance pandemic response by identifying outbreak origins and containment strategies.
- **Environmental and Climate Sciences**:
  - Analyse patterns in moments like bushfires, based on aerial and satellite images.
  - Track and predict climatic events, e.g., cyclones, hurricanes.
  - Study factors affecting various types of pollution using sensor data.
- **Emergency Management**:
  - GIS tools to analyse data from drones, satellites, and social networks for quick responses during crises, from hurricanes to local emergencies.
- **Oceanic and Maritime Activities**:

  - Use spatio-temporal datasets for safe maritime navigation, cargo shipping, and advanced weather forecasting.
  - Monitor environmental changes and detect anomalies like smuggling.
  - Ensure the efficiency and safety of global trade and manufacturing supply chains.

- **Transportation and Urban Planning**:
  - Optimise transportation services like Uber, leading to reduced traffic congestion.
  - Drive urban planning and design based on data from sensors, vehicles, and people.
  - Support the shift towards intelligent, driver-less transportation systems.

- **Agriculture**:
  - Implement precision agriculture techniques using data from drones and sensors.
  - Maximize crop production by optimizing farming techniques based on soil and climate conditions.
  - Address the challenges of decreasing arable land and climate-induced threats to farmland.

- **Emerging Fields of Study**:
  - Monitor and study phenomena like animal migration, space exploration, and neuroscience.
  - Opportunities in non-traditional areas, such as biology, chemistry, and astronomy.

Conclusion: Spatio-temporal databases are a powerful tool for understanding and tackling complex global challenges in a variety of applications. We may rely on data-driven solutions that are both exact and effective in addressing problems by combining these databases into diverse applications and sectors.

2.4.1 MobilityDB

MobilityDB is a modern database system designed specifically for managing and analysing spatio-temporal data, especially movement and trajectory data. MobilityDB, built as an extension to PostgreSQL and its spatial module PostGIS adds specific features to efficiently manage large amounts of spatio-temporal data. Before delving into the reasons why MobilityDB was chosen as the primary database for this thesis, it is crucial to understand some of its main data types, especially those relevant to trajectory prediction:

MobilityDB is a highly adaptable database that benefits from a strong type system that can manage temporal data and track value changes over time. The base type and the time type are the two most important components of this system. The base type can represent a variety of numerical as well as spatial points, but the time type determines how time is represented, such as exact timestamps or larger time periods. It provides a collection of abstract data types (ADTs) designed for moving object data. For instance, `tgeompoint` is a temporal type that encapsulates the movement of a geometric point across time, similar to recording a pedestrian's trajectory point by point. Combining these trajectory points into one sequence to represent a full trajectory can be achieved by combining them into a `tgeompointSeq`. Another example is `tfloat`, on the other hand, tracks the evolution of a floating-point value over time, essentially representing features such as a pedestrian's walking or running speed. In essence, MobilityDB provides a diverse set of temporal types, such as `tgeompoint, tgeogpoint, tint, tfloat, tbool`, and `ttext`, which are smoothly mapped to their PostgreSQL and PostGIS equivalent base types, such as `geometry`(point), `geography`(point), `int`, `float`, `bool`, and `text`[16].

MobilityDB excels in data compression thanks to delta encoding. This method takes advantage of sequential data similarities, making it effective for moving geometries. Furthermore, the interpolation mechanism in MobilityDB detects value transitions between timestamps and presently supports both step and linear interpolation to handle both static and dynamic data circumstances [17].

3 Related Work

The related work section is organised to give a thorough overview of the research that has been already done on trajectory prediction. We begin by talking about rule-based approaches, which use domain-specific rules and heuristics to forecast trajectories. These techniques can be used for certain purposes, including the modelling of pedestrian behaviour or predicting traffic flow, and they often rely on hand-crafted rules based on geographical and temporal information. We also investigate expert systems and decision trees for trajectory prediction, which are rule-based approaches that make predictions using decision trees or knowledge-based systems.

We proceed to explore non-machine learning techniques. Statistical techniques like time series analysis and Markov models, physics-based models like kinematic and dynamic models, and analytical approaches like Gaussian processes and Bayesian networks. We discuss the benefits and drawbacks of these methods in the context of trajectory prediction and offer details on their practicality and constraints.

Then we delve into machine learning techniques, which have recently attracted a lot of interest for trajectory prediction. These techniques use machine learning algorithms to forecast the future by learning from prior trajectory data. We look at supervised learning techniques that use labelled data to train predictive models, including k-nearest neighbours, decision trees, and support vector machines. We also address unsupervised learning techniques that do not require labelled data but group similar trajectories based on similarity indices, such as clustering and density-based techniques. We investigate trajectory prediction approaches based on reinforcement learning, where agents are trained to decide how to anticipate trajectories based on incentives and feedback. We also examine deep learning techniques that have shown promise in capturing the temporal and spatial connections in trajectory data, such as recurrent neural networks (RNN), convolutional neural networks (CNN), and transformers.

We review spatio-temporal data mining techniques in addition to machine learning techniques for trajectory prediction. These techniques make use of data mining tools to find trends and connections in trajectory data.

3.1 Prediction Categories

A significant amount of work and research has been conducted within the field of trajectory prediction. A trajectory is typically described as a sequence of locations arranged by timestamps. Within this field, three main categories are the most frequent:

- Route Planning
- Long-range prediction
- Short-range prediction

Route planning is the process of building a complete route that travels between two points on a geographic location, where the first point serves as the starting point and the second as the destination [18]. All the intermediate trajectories together create a route. There exist various applications of route planning such as finding the most popular or the fastest route between 2 locations.

Long-range predictions attempt to forecast an object's future trajectory over a wider time horizon, such as that of a person or a vehicle. They are different from route planning in that the aim is to forecast a moving object's entire future trajectory. The sequential positions or motions of the object over a longer period of time, frequently several minutes or even hours, are predicted as part of these forecasts, which go beyond simple route planning [4]. The object's velocity, acceleration, interactions with the surroundings and potential future changes in the environment must all be taken into account when making long-range forecasts. Naturally, the uncertainty of this information will cause the prediction accuracy to decrease as the prediction time interval increases. An example where this is compulsory is aircraft trajectory prediction. Zeng et al. [19] mention that the prediction accuracy will inevitably drop as the predicted time interval gets longer due to the complexity and unpredictability of this information.

Short-range predictions have already been widely investigated, where the main concern is the prediction of only the next location. Some of these techniques exclusively use individual movements for predictions, while others take into account all the moving objects' historical movements [20].

The above-mentioned prediction strategies predict locations using past trajectories, however it is obvious that when external data is utilised, it might increase the overall accuracy of the prediction. These can vary from the length of trajectory, travel time, accident reports and road work, which are then used to calculate the likelihood of the predicted locations in the related context [20]. Similar to this, in order to increase the accuracy even more of the prediction model, context data such as time of day, day of week, and velocity have been included as features [21].

3.2 Models and Techniques

As mentioned before, in the area of trajectory prediction, numerous distinct approaches, including machine learning (ML) and non-ML methods, have been proposed. Understanding these current approaches, together with their benefits and drawbacks, is essential for gaining insight into the state of the subject today. In this section, a distinction will be made between non-ML and ML methods, and then we will look into a couple of methods within each category. The following techniques will be discussed:

- Non-ML
  - Rule-based techniques & heuristics
  - Markov modelling
  - Graph-based mining (GBM)
  - Link prediction
- ML
  - Markov modelling
  - Graph neural network (GNN)
  - Long short-term memory (LSTM)

3.3 Non-ML Approaches

3.3.1 Rule-based Techniques

Heuristics and rule-based methods are two different types of trajectory prediction techniques that base predictions on known rules or patterns. These methods are based on the idea that certain patterns or rules can be discovered in trajectory data and used to predict future locations or routes. For example, it is expected that a person who continues to travel between two points A and B will continue doing so in the future.

PrefixSpan is a sequential pattern mining algorithm, with a goal to uncover frequent subsequences as patterns in a database. Morzy proposed a two-step process to mine frequent trajectories from a large dataset. First, trajectory generation is performed and followed by trajectory clustering.

In the first step, sequential patterns are created by repeatedly adding new components from the dataset to a sequence's prefix, then calculating how frequently these extended patterns occur. The candidate trajectories, which are further processed in the trajectory clustering step, are represented by the frequent patterns recovered by the *PrefixSpan* method.

In the second step, a clustering algorithm is applied to group similar trajectories together based on their spatial similarity. It's important to not that even if common trajectories do not perfectly match in terms of movement patterns, the clustering stage aids in finding those that do. This makes it possible to record different object movement patterns while still detecting common ones.

After clustering the frequent trajectories, the writer applied a rule-based method for location prediction. The rules that capture the spatial interactions between the various locations are defined using the frequent trajectories. Based on an object's present location and historical trajectory, these criteria are then applied to forecast its future location. Using these created rules, it is possible to offer insights into the spatial behaviour of objects and the underlying patterns of movement, where the rule-based approach will then produce interpretable predictions. The *PrefixSpan* algorithm and rule-based techniques work together to produce precise location predictions and allows for scalability for larger datasets [22].

It is crucial to remember that this whole methodology depends on the quality of the dataset used and how representative it actually is. There also still might be shortcomings of rule-based techniques in capturing even more complicated and dynamic movement patterns.

In the following example, a hybrid prediction model is proposed. Jeung et al. [23] opted for a combination of a rule-based approach together with a machine learning approach. Association rules are used to capture spatial relations between different locations, which is based on historical data. The Apriori algorithm, which is an algorithm for frequent itemset mining, is used to extract these association rules from the trajectory data. These are then used to perform predictions on the future location of an object based on its current location and their past trajectory (historical data). The crucial difference with the previous example is that a decision tree is used to model the movement patterns. This tree is trained using previous trajectory data of the object. The predictions are then performed based on other attributes such as speed, current location, direction and more [23].

The last study that will be discussed, regarding rule-based techniques, aims to perform trajectory prediction for connected vehicles to create precise traffic management and route optimisation systems. The Prefix-Projection technique is an extension of the previously-mentioned *PrefixSpan* algorithm used for mining frequent patterns. Three essential steps are presented in the algorithm *PrefixTP*. In order to communicate and exchange traffic data, connected vehicles equipped with sensors form a grid of vehicles and produce a large amount of spatio-temporal data. Second, the algorithm creates patterns of increasing length in this stage by projecting the shortest, most frequent patterns onto longer patterns. In order to ensure that only the most relevant and significant patterns are used for prediction, the algorithm prunes patterns that do not reach specific support and confidence requirements. The final step is trajectory matching. In order to determine how accurate the predictions are, the common patterns created in the previous stage are projected onto future time periods. These predictions are then compared to the actual data to be able to evaluate their accuracy [24].

Each of the three studies presented a different approach to trajectory prediction using rule-based techniques in combination with other existing technology. In general, rule-based systems can be efficient for heuristic real-time trajectory prediction since they are able to manage big and complicated data and produce precise predictions. However, the accuracy and power of these strategies rely on the quality and representativeness of the dataset utilised for mining common trajectories. A disadvantage for real-time applications is that rule-based approaches may be computationally expensive. Machine learning approaches, on the other hand, can assist in taking into account new features and reacting to evolving data patterns. They may not function well with small datasets, and they require a lot of training.

3.4 Markov Modelling

A Markov model is a mathematical tool that can be used to estimate the likelihood of a future event. A Markov model can be used to forecast where someone will travel next, based on their current location in the context of location prediction.

An approach that uses Markov models to improve location prediction is the *Next Location Predictor with Markov Modelling* (NLPMM). The system aims to predict the movement of objects by analysing both individual patterns and group patterns over time. This is done by using two separate models, namely the Global Markov Model (GMM) and the Personal Markov Model (PMM). The GMM goal is to analyse the general movement patterns in all trajectories, while the PMM focuses on the individual patterns of each moving object using the historical data that is available. These are merged together using linear regression to perform predictions. Furthermore, the authors take into account that movement patterns or habits can vary depending on the time. For instance, a user might have different habits during weekdays than at the weekend. This is addressed by grouping up similar time periods together and creating a separate model for each group. To evaluate their solution, they tested the approach on real-world data consisting of real vehicle trajectory data. To conclude, NLPMM outperformed existing state-of-the-art methods and proved to even be effective with sparse data [4].

3.4.1 Graph-based Mining

Lee et al. [25] present the Graph-Based Mining (GBM) algorithm as an effective method for mining common trajectory patterns from spatio-temporal databases. Due to its ability to include both spatial and temporal features, the GBM technique is important for applications that require taking both dimensions into account. Every vertex in a mapping graph has a Trajectory Information List (TI-list), which is an important component that the GBM algorithm uses to log all trajectories that pass through it. Using the information in TI-lists, a depth-first search is performed on the mapping graph to extract frequent trajectory patterns. This approach makes use of the adjacency property of the mapping graph to reduce the search space and computational complexity and cost, making the algorithm more efficient and scalable compared to the existing Apriori-based and PrefixSpan-based algorithms, which were mentioned in the previous section. Nonetheless, the authors mentioned several limitations of their approach, including the inability to mine patterns formed by non-adjacent points and challenges in handling TI-lists that are too large to fit in main memory. Despite these drawbacks, the GBM algorithm outperforms competing approaches in terms of efficiency and scalability and provides a new perspective on the analysis of spatio-temporal data.

3.5 Time Series Analysis

The Holt-Winters Exponential Smoothing method is particularly used for time series data that displays both *trend* and *seasonality.* In this specific context, Kalekar [26] refers to 'trend' as the long-term increase or decrease in data and refers to 'seasonality' as changes caused by seasonal factors, such as fluctuations in demand during holidays. The method consists of two distinct models, namely the *Additive* and *Multiplicative model.* The former is used when seasonal changes remain relatively constant over time, while the latter is utilised when seasonal changes are reflected by a certain factor. The following example is given by the author to further explain the difference: When toy sales increase by a million dollars every December, the seasonality is additive. If the sales increase by a certain percentage, e.g. 40%, the seasonality is multiplicative. The model proposed by Kalekar uses three parameters. These are initially set on a specific time period and are constantly updated as more recent data becomes available. This is very important, considering it allows the model to adapt to changes in time series behaviour, which leads to an increase in prediction accuracy. The *Mean Absolute Percentage Error* (MAPE), which determines the average percentage difference between actual and predicted values, is used to assess the model's accuracy. Additionally, the technique also makes use of a certain *look back* size, which stands for the amount of past data that is considered when performing a prediction. This can help in identifying long-term trends and therefore improving the prediction accuracy. In conclusion, the Holt-Winters Exponential Smoothing technique proves to be effective and helpful in adjusting to changes in time series behaviour and improving the prediction accuracy, especially when employed with a large look back size, especially when a large look back size is used.

Although this technique was not discussed in the context of trajectory prediction, its adaptability proves its importance as it allows the model to take into account both old and new habits, which is essential for accurate trajectory prediction.

3.5.1 Link Prediction

Link prediction is an important task in network analysis, where the goal is to identify missing links or even predict future links based on the network's observed structure. It has a wide range of applications, from recommendation systems to bioinformatics and social network analysis. Traditionally, link prediction methods observe the network as static, using node similarities at a specific point in time to predict links. The issue is that some networks, like social networks, are dynamic and contain vertices and edges that may change over time due to changes in the underlying social structure [27].

For instance, Figure 2 shows a depiction of the temporal link prediction problem. Divakaran and Mohan [28] define it as follows: Consider a dynamic network G = (V,E) where V represents the set of vertices and E the set of edges. Changes in the network are captured at specific moments in time, resulting in a series of snapshots G1, G2, ..., Gt, for each time step from 1 to t. The goal is to use these snapshots to predict the structure of the graph at the next time step, which can be seen on the right-hand side of the figure at T+1.

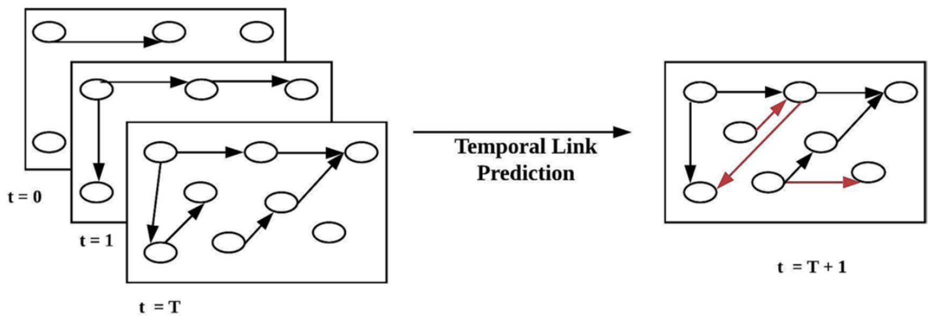

Figure 2: Temporal link prediction goal [28]

Fang et al. [29] propose a dynamic approach to predict links and model individual behaviour in temporal social networks. This is achieved by utilising a method that looks at the network at several points in time rather than just at one moment. We refer to a 'snapshot' of a network as a photo taken of all the connections at a specific moment in time. Traditional methods usually predict links based on how similar nodes are in one of these snapshots, but the approach suggested by the authors is a type time series analysis. By making use of various snapshots, they can better comprehend how the network changes over time. In order to make accurate predictions, the authors embedded the network's graphs into a space that can capture these changes. Performing manifold alignment, which is a process that aligns each snapshot of the network over time, helps the model to capture the network's evolution and thus improve its predictions. Through a number of time series experiments, they were able to show the importance of this alignment.

3.6 ML Approaches

3.6.1 Markov Modelling

Markov models can be also used and enhanced by the use of machine learning. A method created by Ashbrook and Starner used GPS data to classify locations into meaningful places, which were then added to a Markov model. They were able to predict someone's future whereabouts by training the algorithm on their past movements. It is noteworthy that the authors discovered that their method for position prediction demonstrated strong generality across various users. However, they had not yet included time prediction in their Markov model at the time. This means the variable of time, which is also an important factor, was not taken into account in the prediction. In order to support time prediction, they intended to expand the model and look at how variations in arrival and departure timings can reveal the significance of certain occurrences [30].

3.6.2 Graph Neural Networks (GNN)

A specialised class of neural networks, named graph neural networks, offers the capability of predicting which two nodes in a network are likely to be connected by an edge in the future. A solution to this link prediction problem can be applied to many different use cases, such as social recommendation systems and security. With such networks, it is vital to understand the nature and context in which they occur. This means understanding which entities in the network in question have the ability to create new links with other entities in the future. Reliable and accurate link prediction plays a crucial part in our understanding and analysis of the complex networks in which they occur.

A powerful link prediction approach that employs a deep learning framework Capsule Networks (CapsNet) was proposed by Liu et al. [31] CapsNets are able to learn graph structures by leveraging their ability to recognize spatial hierarchies and relationships in complex data. Naturally, combining this with the capability of GNNs to process this graph-structured data provides us with a powerful new tool.

In contrast, earlier projects would tackle this link prediction problem more heuristically. Whilst these methods are effective in practice, they rely heavily on making some strong underlying assumptions about the existence of certain links. As an example, the Common Neighbours technique makes the assumption that two nodes are more likely to be connected by an edge if they share a lot of common neighbours. Due to its relative simplicity and potential for underestimating the complexity of the overall network topology, this approach may only be appropriate for a limited number of situations.

To go a little more into detail, we briefly compare Capsule Networks with a traditional convolutional neural network (CNN). While CNNs represent their features using scalar values, CapsNet opts for vectors. Vectors make it possible to collect additional information about various characteristics of a specific entity. CapsNets also contain pooling layers that lead to information loss, whereas CapsNets use capsules. Capsules are a group of neurons that work together to capture the various properties of a certain entity. To decide which capsules in one layer should be connected to which capsules in the following layer, CapsNets utilise a routing method. This effectively allows for capturing part-whole relationships between entities, which are ideal for object recognition and image segmentation tasks.

To forecast links in complicated networks, Liu et al. offer a two-stage architecture that combines the advantages of GNNs and CapsNets. They learn node embeddings that capture the structural characteristics of a network in the first stage with the GNN. In the second phase, they learn feature embeddings that model the attributes of the nodes using CapsNet. The outputs of the two different networks are then combined in order to make link predictions. According to the results of their tests, Liu et al.'s strategy is around 20% more effective than earlier, more traditional approaches [31].

3.6.3 Long short-term memory (LSTM)

To represent sequential data, including time-series data, text, and audio, the long short-term memory (LSTM) neural network architecture is used. Since Hochreiter and Schmidhuber first introduced it as a technique for modelling sequential data in 1997, it has gained quite some popularity [32]. The LSTM network is one type of neural network that can learn and duplicate extended sequences. They have succeeded with various sequence prediction problems, including speech and handwriting production [33]. Since LSTMs are capable of accurately capturing complex correlations between current and future events, they have recently been used in multiple articles to predict future trajectories.

In the research of Alahi et al. [33], a new approach is proposed to predict human trajectories in crowded scenes using *Long short-term memory* (LSTM) networks. Their approach aims to forecast the trajectories of all individuals in a scene by taking social norms and common sense guidelines into consideration that people usually follow when navigating in communal spaces. The authors suggest employing one LSTM for each person in a scene to predict their whereabouts in the future. The LSTMs share weights, enabling the model to identify patterns that appear frequently in the paths of various individuals. The authors create a Social pooling layer that distributes data among the LSTMs to take into consideration interpersonal interactions. This allows for spatially proximal sequences to share their (hidden) states with each other. In essence, this means that the model is able to pick up on common interactions across trajectories that overlap in time. As an illustration, if two people are walking side by side, their trajectories are likely correlated, and the Social pooling layer enables the model to capture this correlation.

On two publicly accessible datasets, the suggested model, dubbed "Social" LSTM, beats state-of-the-art approaches. Additionally, the authors demonstrate qualitatively how their Social-LSTM effectively anticipates a variety of non-linear behaviours resulting from social interactions, such as a group of people travelling together.

This technique may be used in autonomous cars because it is essential to predict human trajectories for secure navigation precisely. By training on datasets with various moving objects, the Social-LSTM model might be modified to forecast the trajectories of other moving things besides people [33].

The final LSTM approach we will discuss introduces a new approach to pedestrian trajectory prediction in crowded places. The technique, known as *Bi-Prediction*, uses a bidirectional LSTM architecture and distinguishes itself from existing approaches by taking the intended destinations of pedestrians into account rather than focusing on the influence of neighbours. A 'neighbour' is referred to as "neighbouring pedestrians" who could influence a pedestrian's trajectory. There are two stages to how the Bi-Prediction approach works. First, it divides observed trajectories into a variety of route groups. The model then creates numerous trajectory predictions with varying probabilities towards various destination places. Evaluations against two baseline and three cutting-edge approaches on benchmark datasets show that this two-stage strategy improves trajectory prediction accuracy and lets the model generate numerous prediction trajectories with different probabilities. This feature is particularly useful and applicable for situations where anomaly detection and space planning is of importance. Despite its benefits, the current Bi-Prediction implementation requires manual region partitioning identification in the first stage. The authors intend to overcome this issue in their future work by automating this. Additionally, they intend to consider more contextual factors, such as people standing in groups and pedestrians pausing at traffic lights, to improve the prediction accuracy. By using a bidirectional LSTM framework to construct several prediction trajectories for various destination locations, which surpasses existing approaches, the Bi-Prediction method represents a significant advancement in trajectory prediction [34].

3.6.4 Spatio-Temporal Graph Attention Network

While the previous paper focused on adding social interactions into trajectory prediction, the following research takes a somewhat different approach. Huang et al. [35] presents a spatio-temporal graph attention network (STGAT) that incorporates spatial information by generating a graph representation of the environment while using LSTM to simulate the temporal component of trajectory prediction. Both authors recognise the importance of LSTMs in properly predicting trajectories, and this study builds on this basis to increase the model's overall performance.

STGAT employs a sequence-to-sequence architecture, which is a type of neural network that accepts a sequence of inputs and generates a sequence of outputs. The inputs in this scenario are pedestrian trajectories across time, while the outputs are predictions on their future trajectories.

A graph attention network (GAT) layer is combined with an additional long short-term memory (LSTM) layer in the design. At each time step, the GAT records spatial interactions among pedestrians by assigning priority to neighbouring walkers based on their relative locations and velocities. The LSTM records temporal correlations between interactions across time by acquiring the continuity of interactions. By combining these two layers, STGAT is able to model both spatial and temporal interactions between pedestrians, leading to more accurate predictions of their future trajectories.

STGAT has various benefits over other approaches for modelling spatio-temporal interactions. Firstly, it directly models interaction continuity over time, which has not been done before according to my current knowledge.Second, it employs GAT to prioritise neighbouring pedestrians based on their relative locations and velocities, resulting in more accurate predictions. Finally, it outperforms two publicly accessible crowd datasets (ETH and UCY) and generates more "socially" believable pedestrian routes. However, there are some possible disadvantages to employing this method. Because of the usage of several layers and attention techniques, it may be computationally costly. It may also be less successful in predicting trajectory in extremely dynamic or unexpected conditions [35].

3.7 Overview

After having analysed various systems, models and techniques, we must carefully weigh our options for the approach that we will choose. As mentioned before in the introduction, we have decided not to use machine learning-based techniques for several reasons. We must gather and analyse lots of data over a long period of time and at various intervals. Furthermore, for the evaluation it is necessary that we integrate new data to be able to continuously test, experiment and evaluate the model's performance. Machine learning, especially those that involve deep learning architectures like LSTM and GNNs, requires lots of data for training, and whenever new data is added, the retraining process becomes really time-consuming. This is not only inefficient but also unsuitable for real-time applications as they require quick adaptation when new data is presented. For the evaluation process, we prefer to rely on heuristics as they are more straightforward and help us identify particular places where the model fails. This allows for a more thorough examination of any potential inaccuracies or errors. On the other hand, machine learning models can be challenging to evaluate due "black box" nature, as mentioned before.

Considering all these factors, we have decided to use a graph link prediction approach with exponential smoothing as proposed by Chen et al. [4]. The benefit of utilising graph link prediction is that it can capture complex relationships between different entities, which can result in a more understandable and precise representation of the underlying habits that effects the trajectory prediction. This aligns with our requirements for efficiency, ease of evaluation, and the ability to continuously incorporate and adapt to new data.

4 Solution

Before delving into the specifics of the methodology, we will first give a brief overview of the various components that can be seen in Figure 3, which represents the basic structured methodology.

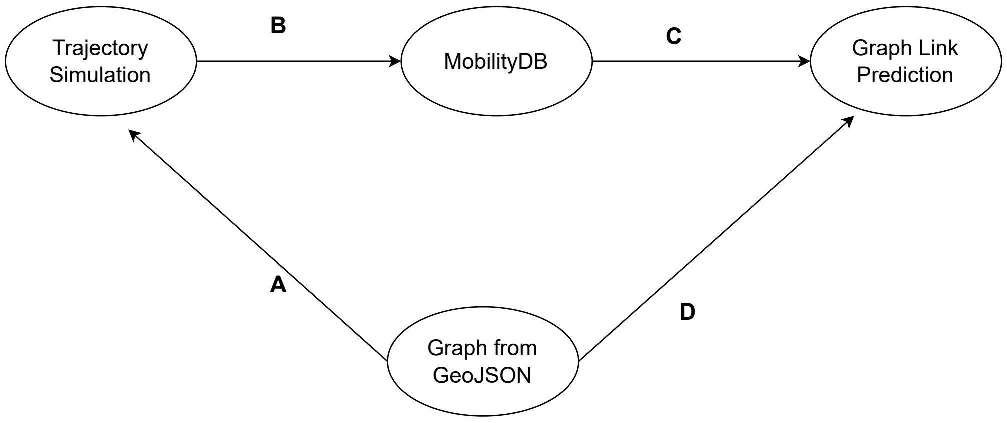

Figure 3: Overview solution

Initially, we create a spatial graph using an existing GeoJSON, which contains the various geographic features of a physical space. Hereafter, we follow arrow 'A', which leads us to the next step, trajectory simulation. The trajectory simulation process uses the GeoJSON-based graph to generate trajectories within this graph. In Section 4.4.3, we further explain how these trajectories are chosen with the use of a persona-driven simulation. The simulated trajectories are then sent into MobilityDB, the database of our choice, as indicated by arrow 'B'. This specialised database has been created to effectively store and manage spatio-temporal data.

As we move into the graph link prediction phase, arrow 'C' shows us the simulated trajectories being used by the graph link predictions. However, it is important to note that the graph link prediction also makes use of the original graph created from the GeoJSON. This means that the predictions are not exclusively based on the simulated habits but also take into consideration the actual environment and infrastructure. By doing this, we ensure that the real-world network's structure and the simulated data are well aligned, which improves the forecasts' accuracy and reliability.

We will start by discussing a few example use cases that the system can handle, as well as the functional and non-functional requirements that must be met to guarantee the system's effectiveness and efficiency.

4.1 Use Cases

There exist many use cases regarding this topic. In a variety of real-world scenarios, especially those that require an understanding of movement patterns, the proposed predictive model would be very useful.

Indoor navigation and route planning: When visiting large indoor spaces such as shopping malls, airports, or exhibition centres, it can be difficult to find the quickest route to your destination. The predictive model can assist in analysing the flow of individuals inside these buildings to determine the places that may become overcrowded at particular periods. This data may be utilised to create interior navigation apps that direct users along the least busy routes, enhancing their experience and easing stress.

Retail stores: The predictive model could be used by retailers to examine client movement patterns and behaviour inside the business. This aids them in understanding the most popular goods and frequently visited areas. Retailers may enhance the shopping experience, increase sales, and better control crowds during peak hours by optimising the store layout in this way.

Resource allocation: Large facilities like universities, hospitals and many other locations frequently require the distribution of resources, including cleaning and maintenance services, security employees, etc. The predictive model can analyse how people move through the facility's various spaces to predict which ones will be crowded or empty at certain times of the day. This information may be used to better allocate resources, such as arranging cleaning services in popular areas during less busy hours or placing security officers in the most needed locations.

While the current version of the model is designed for indoor applications, there are potential extensions of this model for outdoor applications, which will be mentioned below. However, modifications will be necessary to take into account the specific characteristics of outdoor environments.

Urban planning: City planners and transportation organisations can plan public transit routes in a more effective way by analysing commuting patterns and trends. By capturing and learning about areas and their time of peak traffic, it is possible to improve their planning for more effective urban transportation systems.

Traffic management: Whenever a prediction is performed, and it is possible to forecast when a specific area will become crowded, it is possible to offer vehicles an alternative route. Using historical data and trends to predict the traffic flow will lead to better traffic management, which reduces both the overcrowding problem and the travel time of people.

Tourism: Finally, this technique can be used by travel agencies and tourism boards to predict the most popular travel times and routes in the tourism and travel sector. If it is possible to distribute visitor traffic fairly, it could help with improving some tour packages, being able to regulate visitor flow to popular sites and more.

It is also important to note that the real applicability and effectiveness of the model for both indoor and outdoor applications would depend on a variety of variables, including but not limited to the type and quantity of data that is already available and the specifications of each use case.

4.2 System Requirements

In this section, the different requirements for the solution will be mentioned. We start with the non-functional requirements, followed by the functional requirements.

4.2.1 Non-functional Requirements

Here are the non-functional requirements for the system.

- Accuracy: Based on the historical data and contextual information that is provided, the trajectory prediction system should produce accurate predictions. It is one of the most important criteria for assessing the effectiveness of the system.
- Performance: It is crucial for real-world applications that the system will need to process enormous volumes of data and complete computations quickly.
- Personalised prediction: Predictions that are specific to each individual user are produced by the model using personal individual habits that it has learned from past data. Such a method could be useful for planning, for instance, transportation and urban planning, and personalised advertisements by predicting what the user's future location might be.
- Generalisability: The generalisability of the solution is primarily due to its core concepts and techniques, rather than its specific implementation. The use of geospatial and temporal data, the graph-based approach, the learning of personal habits, and the incorporation of semantic context are the fundamental principles that are broadly applicable across a variety of contexts, even though the precise parameters and model specifics may need to be adjusted depending on the use case.

4.2.2 Functional Requirements

These are the functional requirements for the system:

- Data management: The system must be capable of storing user trajectory data in a structured format as well as accepting, processing, and storing that data. This consists of information about the user's beginning and ending locations, their path, and any relevant contextual data.
- Data Generation: The system is able to generate synthetic data based on a user their preferences and/or habits. This includes creating believable trajectories and simulating their habits for various personas.
- Contextualising Trajectories: Based on the time of day (morning, noon, afternoon, evening) and other semantic context information, the system must be able to contextualise trajectories. This contextual information has a significant role in trajectory prediction.
- Temporal Information Management: The system is built in a way that it has to deal with the temporal information that is inherently present in the trajectory data. This involves creating a mechanism which considers the sequence and timing of the trajectories, either when generating new ones or when the predictions are performed.

4.3 Database Schema

MobilityDB is an extension of PostgreSQL and PostGIS, designed to efficiently handle spatio-temporal data. The schema given in this section outlines the relationships between the different entities and attributes involved in the simulated trajectories for the graph link prediction.

- **id**: This is the primary key for the 'user_trajectories' table. The column is auto-incremented (via 'nextval') and guarantees a unique identifier for each trajectory record in the table.
- **userID**: This is another integer field corresponding to the user's unique identifier whose trajectory is being stored. This allows us to link trajectories to specific users and makes it possible to study individual movement patterns. It is important to have a separate 'userID' as a user can have multiple trajectories, and this field allows us to associate each trajectory with the correct user.
- **Trajectory**: This column stores the trajectory data as `tgeompoint` type, a temporal geometric point type in MobilityDB that allows us to store and manage spatio-temporal data. This field essentially captures the user's movement over time and space. The data in the column actually consists of `tgeompointseq`, which contains the datapoints in the form of `tgeompoint`.
- **Semantic_context**: This JSONB type column holds metadata associated with each trajectory, such as the time of day the trajectory was recorded, which can be used in analysis and prediction tasks.
- **Additional_info**: This JSONB type column is designed to hold any extra information that might be relevant to a particular trajectory but does not fit into other predefined columns.
- **Metadata**: This is a JSONB (binary JSON) type column that provides additional contextual information related to each trajectory. This could include information like the purpose of the trip, the mode of transportation, and so on.
- **Trajectory_seq_id**: This UUID (universally unique identifier) column provides a globally unique identifier for each trajectory sequence. This can be useful in complex queries or analyses that span multiple databases or need to integrate with external systems.
- **STbox**: This column of type `STbox` (spatio-temporal bounding box) stores the spatio-temporal extent of each trajectory. This is useful for optimising spatio-temporal queries and helps to filter trajectories quickly based on spatial and temporal bounds.

The database schema has been made to effectively store and query user trajectory data, as well as to support the functional needs 4.2.2 of our trajectory prediction system.

4.4 Data Simulation

4.4.1 Graph Construction

In this section, we will illustrate how to create a graph that represents the geographical space, which we will use for the trajectory simulation in Section 4.4.2, as mentioned in the overview.

The graph is built using the places visited by users as well as the relationships between these sites. To provide a complete and accurate representation, the graph creation process takes into account both the geographical and temporal properties of the trajectories. Once the construction of the graph is completed, the following step will be to generate trajectories.

A pre-existing GeoJSON file, a common open-source format for storing a variety of geographic data structures, served as the primary source of geographical data in our study. This file contained information on geographical elements such as points, lines, and polygons, each of which represented a unique spatial property of the physical environment. Including these pre-defined geographical shapes and polygons made identifying the area of interest much more manageable. However, in order to improve the granularity and accuracy of our trajectory prediction model, "checkpoints" were introduced. These checkpoints, which were placed outside of each room or polygon, function as nodes in our physical space graph model, which can be seen in Figure 4.

This design decision was made to provide for a more simplified implementation of graph-based calculations, notably pathfinding algorithms, allowing for more accurate and efficient trajectory predictions. The trajectory prediction model can achieve more accurate predictions by utilising a graph-based model with checkpoints. This is because the model can utilise the graph structure to calculate optimal paths between checkpoints, considering factors such as distance, obstacles, and constraints. This enables the model better to capture the spatial context and potential movement patterns.

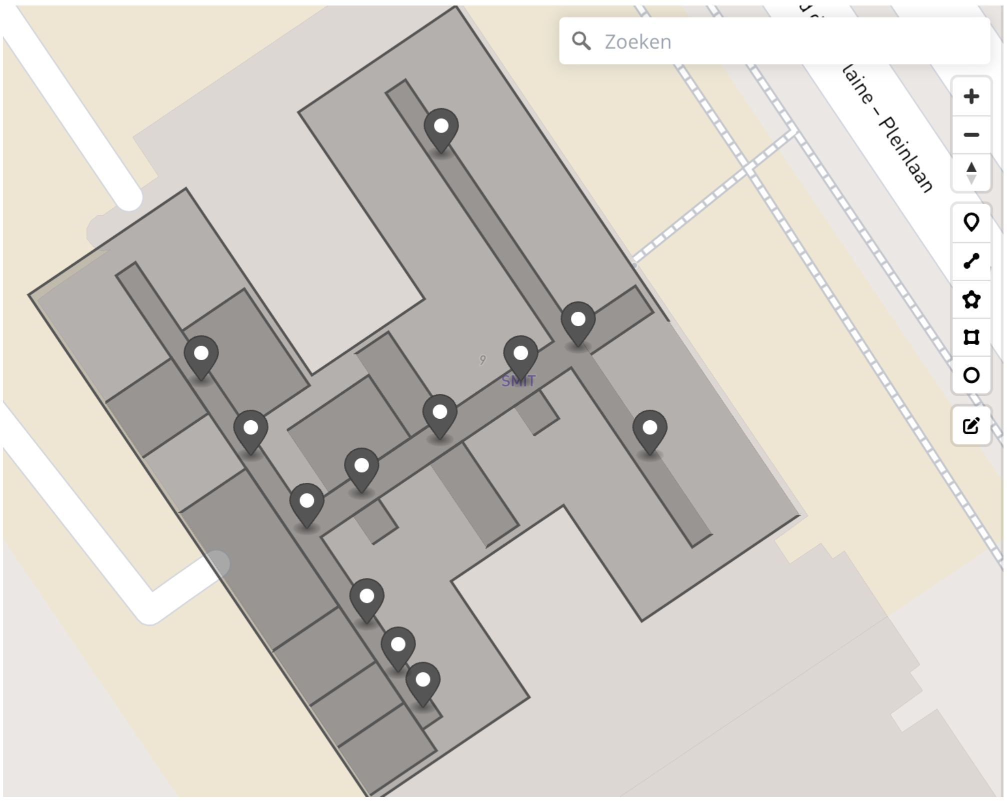

Figure 4: GeoJSON visualisation of Pleinlaan 9, Floor 3 containing rooms and checkpoints

The nodes are represented by the various existing rooms in combination with the checkpoints. In Figure 5, a representation of the graph can be seen with the various rooms and checkpoints as nodes with their corresponding edges.

Constructing the graph in this manner will aid in heuristic trajectory generation, and it has also established a solid foundation for later graph link prediction challenges. Link prediction, in the context of our graph, essentially refers to anticipating future transitions or movements between checkpoints. Our graph is excellently suited for this purpose because of its rich recording of previous mobility patterns, allowing for more accurate predictions of future movements. In order to guarantee that the model accurately captures the real-world constraints, such as avoiding predictions in which users would be able to pass through walls, it is also essential to develop a realistic representation of the physical space. To achieve this, our model is influenced by existing standards similar to, IndoorGML, which is designed to model indoor spaces for navigation purposes [36].

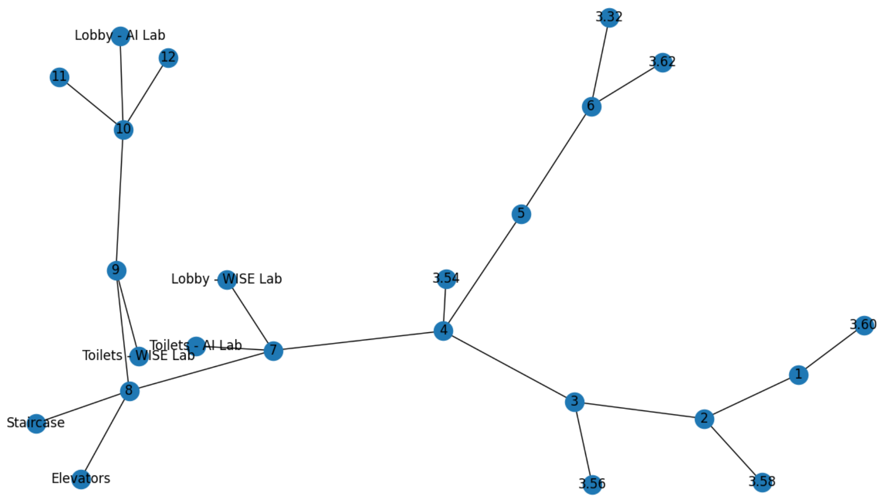

Figure 5: Graph representation of Pleinlaan 9, Floor 3

4.4.2 Trajectory Simulation

The trajectory simulation methodology that we perform, is a comprehensive approach that produces accurate, context-driven, human-like trajectories in a given environment.

Whenever generating a trajectory between a start -and end location, a validation is performed to check whether there exists a path between these 2 nodes. After the verification, the shortest path is computed using Dijkstra's algorithm. The trajectory is built by moving from one node to the next following the calculated path. To offer a more complete depiction of the trajectory, intermediary points, known as sub-nodes, are constructed for each edge along this path. The reason for interpolation is to have a real-world representation of the data being collected. Simply only having the nodes as data points would not be accurate enough, as the granularity level will be too low. Each sub-node indicates the user's estimated position at a specific moment during their trajectory. This fine-grained trajectory not only offers a comprehensive spatial representation of the user's movement, but also includes the trajectory's time element within the spatio-temporal model. This also allows us to really simulate a real-world experience.

As the detailed trajectory is created, the time is incremented based on the calculated travel time, and the average speed. This temporal information in combination with the mode of transportation, in this case walking, forms the basis of the semantic context for a trajectory point. This is further enhanced by classifying the day into four categories: morning, afternoon, evening and night. This classification allows for a more human-like trajectory by tying the movement of users to time-based contexts. Furthermore, when data would be collected in the real-world, it would also allow for trajectory segmentation by splitting up the trajectories of a whole day into four categories. This could possibly lead to interesting information after further analysation of the data.

The trajectory points that have the same semantic context are then batched together. This batching is critical for maintaining the distinctness of each time period and, as a result, ensuring that the produced trajectory matches to the time-dependent context. Once these batches are complete, they are stored in the spatio-temporal mobilityDB database for later analysis.

An extra feature that was implemented, is the use of spatio-temporal bounding boxes, also called STBoxes. In essence, an STBox encapsulates a trajectory's spatial extent and temporal duration, defining the bounds in which the trajectory exists, in both space and time. It has multiple advantages regarding the analysis of spatio-temporal data and trajectory prediction. By associating each trajectory with an STBox, it becomes easier to both identify and acquire trajectories that intersect with a certain spatial area within a given time interval. This saves the computationally costly process of reviewing every single point on a trajectory, resulting in faster and more efficient queries. Another interesting upside, is the detection of unusual trajectories. Initially, a spatial-temporal bound can be set for a person or group, and any trajectory that falls outside these bounds can be flagged as an anomaly within the data.

4.4.3 Persona-driven Simulation

The final section of this chapter is devoted to trajectory simulation utilising a persona with habits. We define a 'persona', which is a depiction of a user with certain characteristics and behaviours, and explain how these personas are utilised to simulate realistic trajectories. This step entails a thorough knowledge of user behaviour and makes use of the data offered by the trajectories to develop rich, habit-driven personas.

This approach assumes that a person's movements follow a predictable pattern that reflects their daily routine. We focused on modelling an office environment, where each persona represents an office worker with a fixed schedule. The workplace layout was illustrated through a graph as mentioned in Section 4.4.1, which showed nodes representing different locations like individual offices, conference rooms, break rooms, etc. and edges indicating possible routes for movement.

The daily habits of the persona were shown as a series of places they visited throughout the day. Each habit was given a specific time frame when it was usually done. For instance, someone might work at 'Office 3.58' from 9 a.m. to 11 a.m., then take a break in the break room from

11 a.m. to 11:30 a.m., and so on. This predictable sequence formed the basis for each character's movement patterns. It is obvious that in the real world, a person will not always be 100% consistent in their daily behavioural habits.

To make the persona's movements more realistic and unpredictable, we added some randomisation to their habits. Humans tend to follow patterns, but they are not always predictable.

First, each habit's execution time is randomised within a certain time span. For example, a "toilet break" habit may occur around 10:30 a.m. on average, although the specific timing varies. This variability reflects the fact that although people might have a routine, the exact timing can vary from day to day.

To simulate a person's daily decisions, we also use randomisation. For instance, after taking a coffee break at 11:30 a.m., the person may either go back to work or use the restroom. This selection is made with a certain probability. This acknowledges that a person's actions can be influenced by various uncontrollable factors.

Apart from daily habits, there are weekly and monthly habits that are associated with certain days of the week or month. These habits are also prone to variation. Sometimes the day of these behaviours is changed, increasing the unpredictability and thus realism of the persona's movements and habits. Another component of the persona's routine that varies is the introduction of entirely random behaviours, such as visiting a random location or meeting with a random member. These behaviours are triggered with a particular likelihood, so they do not happen every day, but they bring unpredictability to the persona's routine.

The semantic context was integrated into each created trajectory. This semantic context added more information to the trajectory, improving its interpretability and use for future research. It contains information such as the method of transportation (in this example, walking) and the time of day (morning, afternoon, evening, and night), which was dynamically updated depending on the current time during trajectory development.

After having successfully generated accurate trajectory data, we want to implement the graph link prediction. This will be the following topic in Section 5, which contains the full implementation on how the graph link prediction was tackled.

4.5 Graph Link Prediction

In the context of the research, graph link prediction is a technique for forecasting people's future trajectories. This is based on historical data containing people's behaviours and movements in the past. A brief theoretical overview will be given regarding the method, which is necessary before delving into the specifics of how the graph link prediction is exactly used.

A network, more specifically a graph, consists of nodes and the relationships between those nodes are called edges. Whenever predicting a trajectory, the graph is used as a representation of the physical space where all the trajectories and movements occur, and each edge stands for a potential path to take. The goal of graph link prediction is to predict new links/associations between nodes in a network based on already existing links. Naturally, this is very important when we want to anticipate what the following steps will be in a trajectory. If we think of each location in a space as a node and the movement from one location to another as a link between these nodes, then the predicting of future locations of a user is equivalent to predicting new links in the graph.

4.6 Exponential Smoothing

Exponential smoothing is a time series forecasting method. This method gives a greater weight to recent observations while still valuing older ones. The key strength of this method is its simplicity and its ability to adjust to changes over time.

Understanding that not all historical data is equally important is essential when analysing historical data to forecast future trajectories. For instance, a person's behaviour at the current time may not be well predicted based on their movements and habits five years ago. There could be multiple reasons for this. For instance, a user might have different habits on the routes they take to work based on the season of the year. During the winter, they would take the shortest route to work because of the colder weather. During spring, the temperature rises, there is less rain, and the weather is overall better. As a consequence, the user might start taking detours to work because they want to pass by a certain park. This is a very specific example, but there could be changes in anyone's habits for many other reasons.

In Section 5.4, you will see how we have added the idea of exponential smoothing into our method of predicting movements. We have named this the 'decay' feature. We can be more certain that our predictions will be accurate if we make sure they are based on the most recent and relevant patterns. The combination of both past behaviour and current habits is what makes our approach effective.

5 Trajectory Prediction

This section will cover the complete implementation of graph link prediction to perform heuristic real-time trajectory prediction.

5.1 Querying and Processing Data

Whenever performing trajectory prediction, we need to train the model for each user separately. This is necessary considering each user has their trajectories with their own habits included. All the trajectories are queried from a user, but we have to remember they were saved as `tgeompoint`. Naturally, it is not possible to save geometric objects into the database without transforming them into an accepting format. With the use of MobilityDB, which is built on top of PostGIS [17], we are able to accept formats such as Well-Known Binary (WKB) or Well-Known Text (WKT), which are formats used to represent geometrical data, such as points, lines, polygons and more. The data points within the trajectory sequences will be of the following format `[WKB@Timestamp]`. These trajectories are initially split into the datapoints and timestamps, which are then stored in a *GeoDataframe*[1]. This is an extension to the regular dataframe from pandas, but allows for the storage of geometry columns and performing spatial operations on them. Furthermore, the specific '*time_of_day*' that corresponds to values such as 'morning' or 'afternoon' is also kept within the GeoDataframe for each trajectory, which will be of use later.

5.2 Graph Construction

The initial graph that was mentioned in Section 4.4.1, is our basis for simulating trajectories. After successfully querying and processing the data, we map each point along a trajectory to the nearest node on the graph. This allows us to use the network's structure for predictions by lining up the raw location data with the existing network.

Hereafter, the data points of each trajectory are mapped onto the nearest edge in the graph. There are two primary reasons for this. First, recording user trajectories taken in the real world is crucial, as they may not perfectly match network edges. Real-world movement does not always cleanly line up with a simplified network representation, such as a road network graph when anticipating movement paths or mapping out trajectories. There could be several causes for this difference. For instance, a user may deviate off the pavement, jaywalk, take a shortcut, or make an unexpected turn depending on the user's local knowledge, road closures, and other variables. By mapping the points onto the respective closest edge, the route will be as closely aligned with the actual road network as possible. Thus, even if the raw data indicates that a user has strayed off the pavement or has taken a route that doesn't exist in the graph network, we can still extract a trajectory that more closely matches the most likely real-world route followed by the user. Additionally, GPS data, frequently used to acquire these trajectories, is not perfect and, due to the presence of noise, may even indicate a wrong location. Secondly, by mapping the data points onto the edges of the graph, we can assign a weight to the edges on how frequently the user has used them.

[1] `https://geopandas.org/en/stable/getting_started/introduction.html`

Finally, by utilising the frequency of the edge usage by a user after having the data points mapped to the corresponding edges, it is possible to use the frequency as a weight. These weights give information on how often someone uses an edge, which is in our case very useful for predicting future trajectories.

5.3 Assignment Edge Weights

After preparing the graph, the following step will be to assign a weight for each edge. The model can better understand the frequency of travel between distinct nodes by assigning weights to the graph's edges, capturing movement patterns within the dataset.

At first, we have no information on the likelihood of a transition for all edges, considering the initial weight for all edges is zero. After looping through the data points and mapping them to the correct edge, the method will keep count of how many times each edge is used. Each edge in the graph now has a weight consistent with the number of times the trajectories present in the dataset traversed it. The higher the weight, the more frequently that path is used, indicating that it's a common route to travel (for a specific user).

While assigning weights to the edges, we keep track of more information. This will be further discussed in the following Section 5.4.

5.4 Decay

Whenever performing a prediction for a user, it's necessary to update the weights in the graph by *decaying* them. This component adds some type of "temporal memory" to the model by giving more recent data an upper hand over older data. This is based on the idea that people's most recent habits are more informative of their future movements.

```python

def calculate_decay(last_updated, most_recent_timestamp, time_of_day, decay_rate):
    if most_recent_timestamp and time_of_day in most_recent_timestamp:
        time_difference = last_updated - most_recent_timestamp[time_of_day]
        time_difference_in_hours = time_difference.seconds / 3600
        decay = 0.5 ** (time_difference_in_hours / decay_rate)
    else:
        decay = 0.5
    return decay

```

Listing 1: Decay Calculation of Weights

In Listing 1, we can see the method responsible for decaying the weights. The goal here is to decrease the value of the weights using a formula, which depends on the following parameters: `last_updated`, `most_recent_timestamp`, `time_of_day` and `decay_rate`

The first factor last_updated is the last timestamp at which a certain trajectory (or edge) their weight updated. This can occur due to a new trajectory using that edge or a decay process that periodically reduces all edge weights to account for the passing of time. The update does not necessarily mean that the edge was used as it reflects the time that edge's weight was last updated. As time passes, the system's perception of an edge's importance gradually decreases if there is no new data to support it. This decline in weight over time is achieved through the decay process that affects all edges, which the last_updated field indicates. Nonetheless, this process does not entirely remove the impact of older data, instead it is decreased to make it possible for older, but still relevant data to influence predictions.

The following parameters that we are discussing are most_recent_timestamp and time_of_day, which are closely related to each other, as the former is a dictionary and the latter are the keys to access the values within the dictionary. This dictionary contains the latest timestamps where an edge was used for each time categorisation, namely morning, afternoon, and evening. By dividing the day into multiple time periods, we take into account that a user's habits or walking pattern can differ depending on the time_of_day. A path that is frequently used in the morning may not be as busy in the evening. By keeping track of the most recent usage during these different times, we can adjust the weights of the edges to match a person's daily routine better. This approach is based on the work of Chen et al. [4], but with a key difference, namely the further splitting of the day into several time periods, that allows for a more precise and personalised adaption of the model to a person's daily routine This means that for the same edge, we could have different most_recent_timestamp values depending on whether the pedestrian typically uses it in the morning, afternoon, or evening. This makes the model more accurate and better suited to each individual's needs. To sum it up, the combination of last_updated and most_recent_timestamp allows the model to balance the effects of respectively recent overall activity and recent specific usage of each edge.

The last factor we will discuss is the actual decay_rate.

5.5 Decay Rate

The decay rate is a parameter, which allows us to determine the rate at which we reduce the significance of older data. This can be seen as the factor that decides how quickly data should "fade out". In our decay function, the formula used is:

$$decay = 0.5^X$$

where

$$X = \frac{time\_difference\_in\_hours}{decay\_rate}$$

Here, the decay rate can be seen in the denominator. The larger the decay_rate, the smaller the fraction becomes, making the overall decay value closer to 1 (less decay). Conversely, a smaller decay_rate leads to faster decay as the fraction becomes larger, pushing the decay value further from 1 and closer to 0.

We will illustrate the full decay method with a concrete example.

5.5.1 Example

Imagine a user named X who typically takes a walk through Park A to reach their office. Over time, the edge representing Park A accumulates a high weight due to frequent use of the edge. The `most_recent_timestamp` for morning and `last_updated` are regularly updated as she walks through the park each morning. One day, a new coffee shop opens up, and it lies on a different route which does not include Park A. X will adjust their route and start taking the new one to make a stop by the new coffee shop. As X starts taking a new route in the mornings that doesn't include Park A, the edge representing it accumulates less weight due to less frequent traversals.

To understand the effects of time and decay, let's apply a `decay_rate` of 4 to this example. Given time differences of 4, 8, and 12 hours, the decay can be calculated as:

$$decay = 0.5^{\frac{4}{4}} = 0.5^1 = 0.5$$

$$decay = 0.5^{\frac{8}{4}} = 0.5^2 = 0.25$$

$$decay = 0.5^{\frac{12}{4}} = 0.5^3 = 0.125$$

After the edge has not been used for

- 4 hours, the weight has decayed to 50% of its original value
- 8 hours, the weight has decayed to 25% of its original value
- 12 hours, the weight has decayed to 12.5% of its original value

While the `last_updated` for the edge is still being updated through the decay mechanism, the `most_recent_timestamp` for the morning is no longer updated since X no longer uses this route. In this scenario, both factors provide a different insight into the user's behaviour. If we were only looking at `last_updated`, we might overestimate the chance of X taking the Park A route in the morning. On the contrary, if we only took `most_recent_timestamp` into consideration, there would be no account for the decrease in weight in X's morning routes through Park A. By incorporating both the `last_updated` and `most_recent_timestamp` values in our decay calculation, we can create a model that is both adaptive (responding to recent changes in behaviour) and robust (keeping valuable information about past habits). This allows us to offer more accurate real-time predictions, taking into account both the current and past patterns of an individual's walking habits. Furthermore, introducing a `decay_rate` of 8 would significantly adjust the speed at which an edge's weight decays, underscoring the importance of carefully choosing this parameter to capture and predict user behaviour accurately. This will be further analysed in Section 6.

5.6 Temporal Habits

When trying to predict a user's future movements or habits, it is not enough to only consider their physical location. The specific time when they tend to follow certain patterns can greatly improve our predictions. In this section, we will explain how we include the temporal aspect in our predictive model.

Whenever performing a prediction in real-time, the model will take the current time and location into account. The goal is to take create an interval that includes the time of the prediction. If we perform a prediction at 12 sharp, we create, for example, an interval from 11:00 until 13:00. Hereafter, we execute the query that can be seen in Listing 2:

```
1       SELECT
2           atPeriod(
3               trajectory,
4               period(concat(date, ' {time1}')::timestamp, concat(date, ' {time2}')::timestamp)
5           )
6       FROM
7           (SELECT trajectory, date(startTimestamp(trajectory))
8               FROM user_trajectories) AS result;
```

Listing 2: Temporal query

MobilityDB provides us with a way to query all the trajectories between 2 distinct timestamps, with the use of `atPeriod`. However, the current implementation has a drawback: while it can easily retrieve trajectories within a specified date-time range (for example, between `atPeriod(Traj, '[2012-01-01 08:05:00,2012-01-01 08:10:00]')` it is unable to natively filter trajectories based solely on time intervals, separated from the date (for example, between 08:05:00 and 08:10:00 across all dates).

Our goal is to query all the trajectory data that falls within a particular time interval, no matter the date. The workaround for this limitation is as follows: We break down the timestamp to separate the date from the time component. This allows us to dynamically create the exact datetime intervals, which *atPeriod* accepts, for every unique date in the dataset.

To reiterate, we first query the date from the `startTimestamp` of each trajectory (together with the trajectory itself) on lines 7 and 8 of Listing 2. Hereafter, we use the date to construct the full datetime interval by concatenating the acquired date with the time interval values (`{time1} and {time2}`) (line 2). Finally, `atPeriod` will return the trajectories that fall within the constructed datetime intervals.

After acquiring all the trajectories within the desired time interval, the next goal is to count the number of times the start location is followed by another specific node. A dictionary will hold the frequency of each unique transition from the specified start location to any other node. For instance, if the start location/node is *Room 3.58, Pleinlaan 9*, then the following dictionary shows the frequency of each unique transition to another node within the time interval calculated above:

`{{Room 3.60: 8}, {Lobby - WISE Lab: 42}, {Elevators: 62}}`

The dictionary serves as a pattern recognition mechanism, allowing the system to capture and remember recurrent movements. Whenever lines 2 and 3 are executed in Listing 3, it multiplies the already computed probabilities (from the decay function) by the historical transition frequency from the dictionary. If a particular transition isn't found in the dictionary, it will default the value to 0 + 1, meaning the original probability is multiplied by 1, remaining unchanged. We maintain a baseline chance for every possibility by adding 1, ensuring we don't exclude possibilities just because they haven't been observed before. This way, the model increases the probability of places that were previously visited more frequently, encouraging the preference for habits.

```python
    history_counts = find_previous_occurences(time, current_location)
    probabilities = [p * (history_counts.get(str(node), 0) + 1)
        for node, p in zip([edge[1] for edge in edges], probabilities)]
```

Listing 3: Adjust probabilities based on historical data

We will provide an example to clarify how the various elements in the prediction model work together. First, a simplification will be given regarding Listing 3.

$$AdjustedProbability = OriginalProbability * (HistoricalCountforNode + 1)$$

- Original Probability **p**: represents the probability derived from the decay function and the current state of the graph (see Section 5.4)
- Historical Count (`history_counts.get(str(node), 0) + 1`): represents number of times a transition from the current location to the node in question has occurred in the past

This means that for nodes with a **high historical count**, the initial probability will be significantly amplified, while nodes with a **lower historical count** won't experience this as much. Imagine we know the following information: we have 2 probabilities of 30% and 50% with respectively a historical count of 62 and 35.

$$AdjustedProbability = 0.3 * (62 + 1) = 18.9$$

$$AdjustedProbability = 0.5 * (30 + 1) = 18$$

While the second node had a higher chance original probability, the first node had a historical occurrence. This brought the final probabilities much closer to each other, where the first probability is slightly higher than the second.

While it is important to be able to recognise new patterns, it's equally important to respect frequently used paths. The multiplication we perform ensures that habits from the past are not easily overshadowed by new ones.

5.7 Choosing Next location

Once the probabilities for the next location are calculated, we do not randomly select one, but a deterministic approach is used. Whenever performing a prediction, a "top 3" is created, with the three most likely locations to be the next one. Hereafter, the probabilities are adjusted as explained in Section 5.6. This approach provides a more deterministic and stable trajectory prediction compared to a purely probabilistic selection.

It is important to note that we keep track of previously visited nodes within the prediction sequence. This helps us avoid revisiting the same location and ensures that the trajectory is more realistic and in line with typical human movement patterns.

6 Evaluation

The purpose of the evaluation is to measure how accurate our predictive model is in various scenarios. We will start by examining the modifications we covered in Section 5.6, particularly the addition of temporal habits. This will be done by measuring how the prediction accuracy changes when we take the temporal habits into account.

We divide our data into two portions: 80% is utilised to train the model, and the remaining 20% will be used as our test dataset. We will add new habits to the data after we get our initial findings to evaluate how the model responds, as it is necessary to understand how the model will respond to the change of having more recent, new habits. Finally, when using real-world data, there is inherently noise present in the data. We want to see what effect the noise will have on the prediction accuracy of our model. This will allow us to examine how well our model handles imperfect data.

We aim to provide an in-depth analysis of our model's performance and areas for improvement in this chapter.

6.1 Data

In Section 4.4.2 we discussed the trajectory simulation methodology and in Section 4.4.3 the persona-driven simulation. We introduce a persona, which essentially is a user, that has multiple habits that they perform each day. Every time a habit is performed by the user, a trajectory will be generated from start location to the end location. The trajectory is initially represented as path, but by interpolating between nodes we aim to make the trajectory more realistic.

The dataset contains 600 trajectories. For the evaluations underneath, it remains unchanged unless mentioned otherwise.

6.2 Temporal Habit Factor

In our initial implementation of the graph link prediction, we only took the spatial aspect into consideration. However, in Section 5.6, we have incorporated the temporal dimension into our predictive model. The ultimate goal is to detect recurring movements (habits) that occur at specific times. For instance, if a user goes to the *Lobby - WISE Lab* every single day around 12:00 midday, we have to take this into account. By analysing the trajectories around this timestamp, we can extract the historic frequency of all locations the user has gone to from their current position, for example `{{Room 3.60: 8}, {Lobby - WISE Lab: 42}, {Elevators: 62}}`. These frequencies are then used to adjust the probabilities that were returned from the graph link prediction, which can be seen in Listing 3.

We now show the importance of the historical frequency by multiplying using a certain factor which is smaller than 1. This is done by changing line 2 from Listing 3 from

```
probabilities = [p * (history_counts.get(str(node), 0) + 1)
```

to

```
probabilities = [p * (FACTOR * history_counts.get(str(node), 0) + 1)
```

By underscoring the effect of the historical frequency, we aim to show whether it has a positive effect on the predictions.

Table 1 shows 2 different measures of accuracy. The first row shows the percentage of 100% correctly predicted trajectories from the start until the end, which we will refer to as perfect predictions. The second row shows the overall mean prediction accuracy, which contains both perfect predictions and non-perfect ones. On Listing 4, we can see how this accuracy is actually calculated. This function compares each element of the actual trajectory to its counterpart in the predicted trajectory iteratively. The accuracy is then calculated by dividing the number of correct matches by the length of the shorter trajectory. The function quantifies how closely the model's prediction matches the actual trajectory. For clarification, Table 2 contains specific examples.

```python
    def evaluate_predictions(actual, predicted):
        num_correct = sum(a == b for a, b in zip(actual, predicted))
        accuracy = num_correct / min(len(predicted), len(actual))
        return accuracy
```

Listing 4: Accuracy Calculation

From the results, we can see a clear decline when comparing the percentages. The more we underscore the historical frequency, the fewer perfect predictions we have. This suggests that incorporating the temporal dimension and accounting for temporal habits enhances the prediction accuracy of the model.

	1	**1/3**	**1/10**	**1/100**	**1/1000**
Perfect Predictions	73.44%	70.40%	48.80%	43.09%	32.00%
Mean Accuracy	89.51%	87.30%	83.77%	82.67	80.30%

Table 1: Accuracy comparison for various underscoring factors.

Underneath, I have provided 5 images of histograms showing the distribution of the prediction accuracy using different factors to underscore the historical frequency. The x-axis shows the various accuracies (binwidth is 0.1), and the y-axis the prediction accuracy.

Figure 6 shows the distribution of the prediction accuracy when no underscoring is performed, using a factor of 1. It's important to note that the values from *Perfect Predictions* in Table 1 match the percentage of the [0.9 - 1.0] bin for every figure. This is because all prediction accuracies in this bin are exclusively 100%.

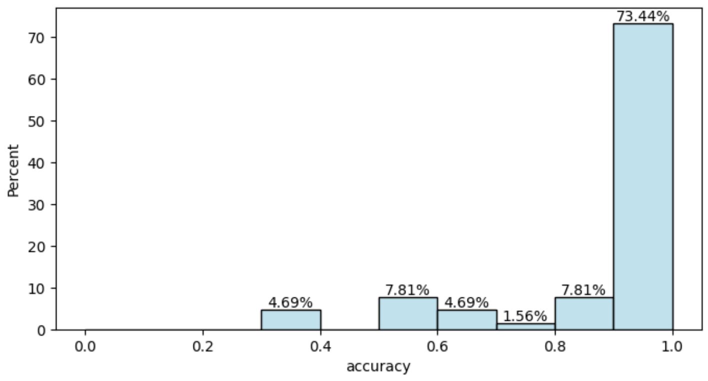

Figure 6: Factor of 1 (no underscoring)

It is important to keep in mind that the prediction model does not predict full trajectories based on a starting location and time. Everytime a prediction is performed, we only know the following singular location the user will be at. By doing this multiple times, we're able to predict a full trajectory. For this reason we have added the Mean Accuracy in Table 1. The percentage tells us how wrong the non-perfect prediction are. Figures 7, 8, 9 and 10 portray the effect of underscoring the historical frequency respectively, with a factor of 1/3, 1/10, 1/100 and 1/1000.

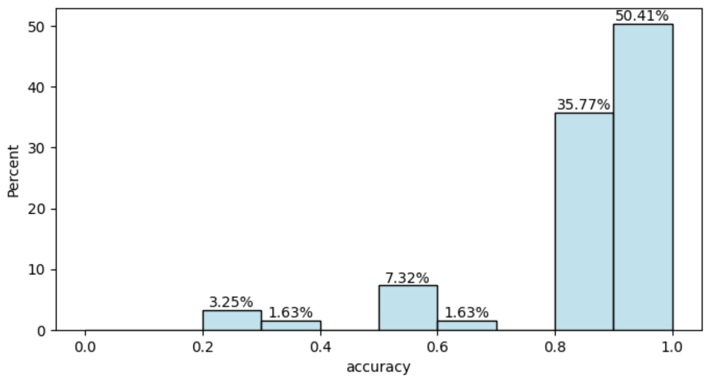

Figure 7: Accuracy Distribution - underscoring factor of 1/3

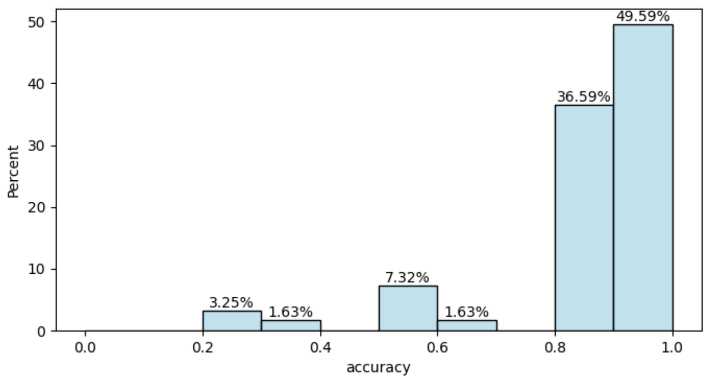

Figure 8: Accuracy Distribution - underscoring factor of 1/10

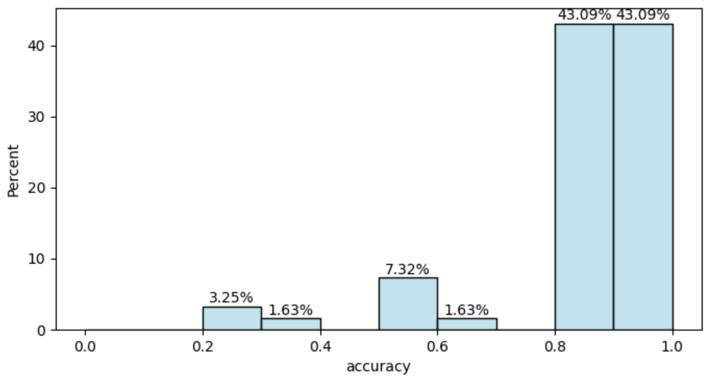

Figure 9: Accuracy Distribution - underscoring factor of 1/100

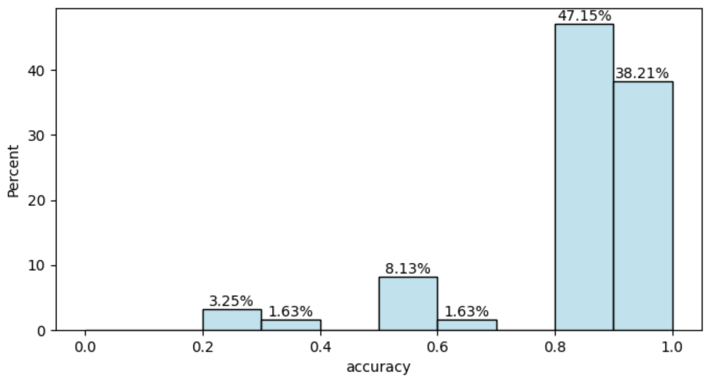

Figure 10: Accuracy Distribution - underscoring factor of 1/000

It is clear on Figures 7, 8, 9 that some of the perfect predictions from the [0.9 - 1.0] bin move to the [0.8 - 0.9] bin, while the rest remains identical. For Figure 10 this is also the case, except that the [0.5 - 0.6] experiences a small increase.

There is a plausible explanation for this phenomenon. Figure 11 displays the physical office space of the user, where also the trajectories are simulated. The middle corridor and right section, marked in a red box, has a lot of rooms and checkpoints, in comparison to upper left area, marked in a blue box, only contains 2 rooms and 2 checkpoints. The same counts for the lower left area, marked in yellow, as this only contains 3 rooms and 2 checkpoints, not including the user's main office 'X' (Room 3.60). The red box is strictly reachable by passing by checkpoint 4. the edge weight between points (4,7) keeps growing with each trajectory, every time a trajectory is generated with the end location in the red area. We previously stated that by including the historical occurrence factor in our probabilities, the amount of perfect predictions increases. The reason the distribution for the rest of these trajectories remains the same is that the weight on edge (4,7) has been accumulated to a certain point that the historical occurrence factor has no effect on the outcome of the prediction. This is exactly what happens for trajectories 1 & 3 on Table 2. Naturally, this also counts for the edges previous to (4,7), namely (2,3) and (3,4) as this can be observed in trajectory 2.

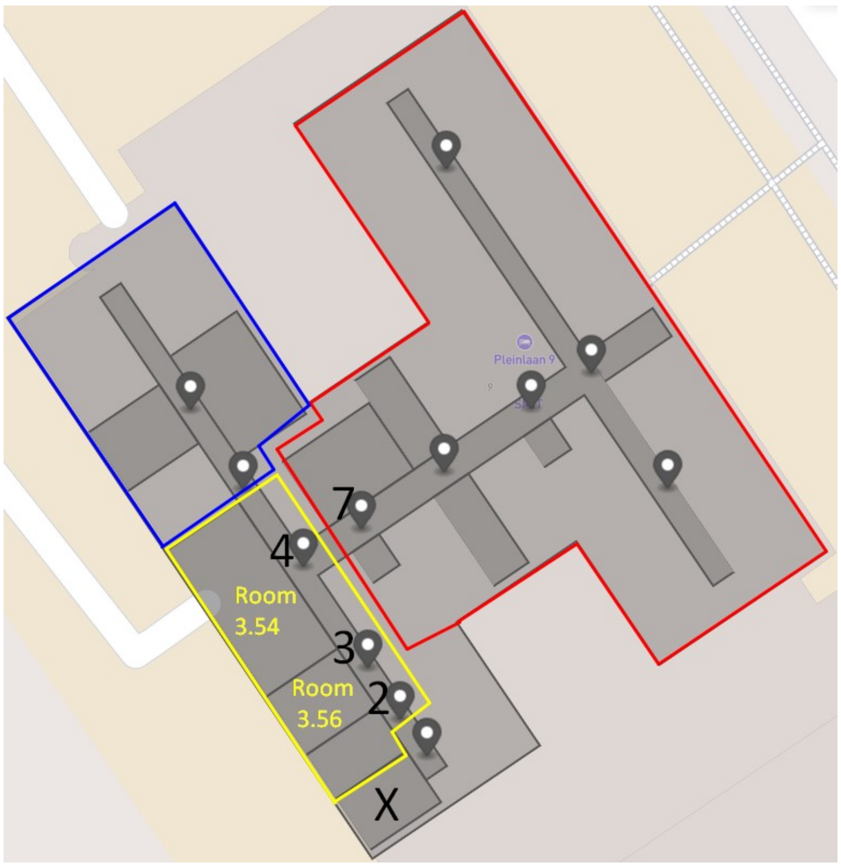

Figure 11: GeoJSON Floor 3, Pleinlaan 9

		Trajectory	Value
Trajectory 1	Predicted	['2', '3', '4', '7', '8', 'Elevators']	
	Actual	['2', '3', '3.56']	
	Start location	Room 3.60	
	Accuracy		0.6667
Trajectory 2	Predicted	['2', '3', '4', '7', '8', 'Elevators']	
	Actual	['2', '1', '3.60']	
	Start location	Room 3.58	
	Accuracy		0.3333
Trajectory 3	Predicted	['2', '3', '4', '7', '8', 'Elevators']	
	Actual	['2', '3', '4', '3.54']	
	Start Location	Room 3.60	
	Accuracy		0.75

Table 2: Comparison of Predicted and Actual Trajectories

Conclusion : The distribution consistency in some histograms is likely due to the physical arrangement and constraints of the office. The red part, with its higher density of rooms and checkpoints, provides more trajectory options than the blue or yellow sections, which have fewer checkpoints or rooms. This has a significant impact on trajectory prediction and accuracy.

6.3 New Habits

In any real-world scenario, habits do not stay the same over time. Depending on a lot of variables, these might change overtime. For instance, imagine a university student who usually takes a shortcut through the campus to reach class during the winter due to the cold weather. However, when spring arrives, they might take a detour because they prefer walking through a park. This is a simple example that shows how habits change over time and how our model should be able to adapt to changes to maintain performing accurate predictions.

The histogram on Figure 6 shows the accuracy distribution for 600 trajectories in our dataset. While the user normally follows reoccurring habits, there are 6 that vary and have a certain chance of occurring. While each habit has a designated time, it can fluctuate, occurring up to 45 minutes earlier or later, ensuring that it's representative of a real-world scenario.

The newly added habits lead to new unique trajectories, but might contain some overlapping intermediate locations with the regular habits. With the use of 6 new habits, a total of 300 trajectories were added.

The histogram shown in Figure 12 shows a small decrease in the number of perfect predictions. This doesn't mean per se that our model is adaptive to changing habits, but it still performs fairly well.

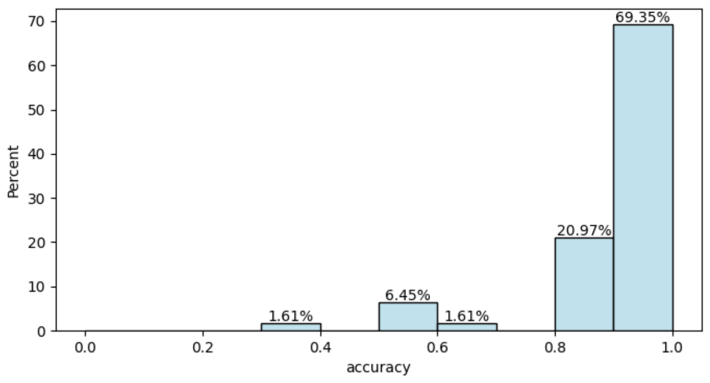

Figure 12: Accuracy Distribution - with new habits

The dataset currently contains 900 total trajectories. We will increase the number of regular habits from 600 to 1800 and the new habits from 300 to 900. This will lead to a larger dataset, where the difference between the regular and new habits is substantial. The results can be found on Figure 13.

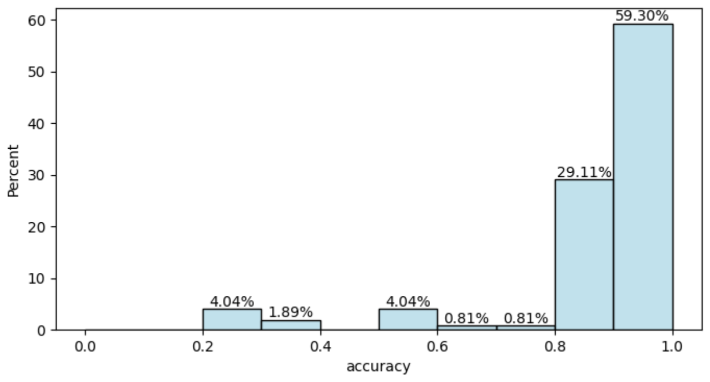

Figure 13: Accuracy Distribution - larger dataset with new habits

The accuracy distribution has changed slightly, but we observe the same effect as in previous Section 6.2. Some of the perfect predictions are lost and transferred over to the [0.8 - 0.9] bin.

6.4 Noise

The current evaluation is performed on purely simulated data. In the real word, positioning systems are usually never perfect in data collection, leading to noisy data. According to [37] et al, the average noise in positioning systems (especially indoor), is between 1-3 meters. This is very important considering our current solution relies solely on perfect, noiseless data. In this section, we aim to evaluate how well the model handles inherent noise.

To simulate the noise, two different noise levels have been applied to the dataset. Figure 14 shows a variation ranging from -1 to +1 meters and Figure 15 another from -2 and +2 meters.

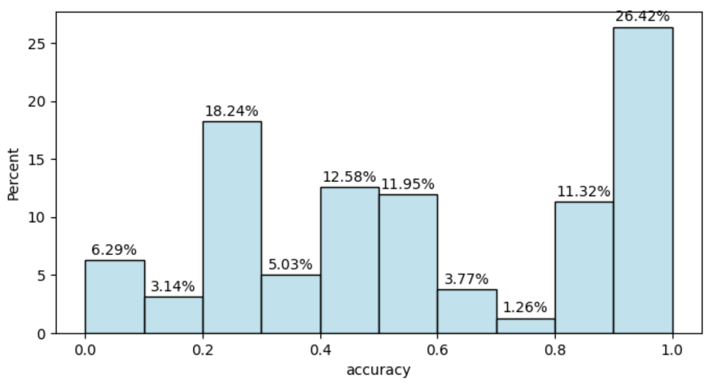

Figure 14: Accuracy Distribution - Noise level (-1,+1) meter

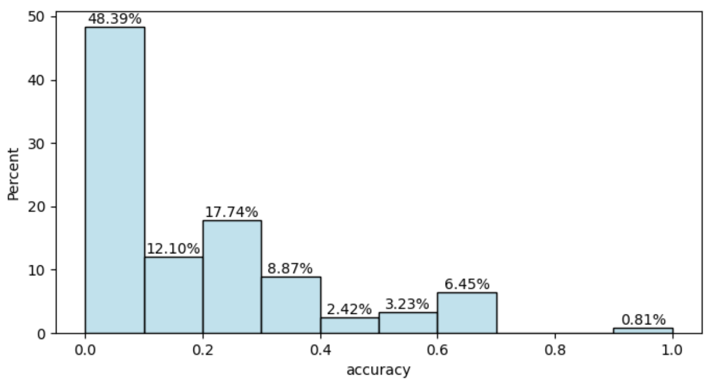

Figure 15: Accuracy Distribution - Noise level (-2,+2) meter

If we compare the accuracy distribution from the previous two Sections 6.2 and 6.3 with the current section, it becomes clear that the model struggles to adapt to the inherent noise present in real-world data. This primarily due to the node mappings applied to the database trajectories as mentioned in Section 5.2. It describes how each point on a trajectory is mapped onto the closest node on the graph.

Because of the layout of the floor and the proximity of points of interest such as rooms and checkpoints, even minor deviations caused by this noise can result in faulty node mappings. As a result of these node mappings, the trajectories created are chaotic and meaningless. The model is limited in its adaptability because of the actual space's compactness, mixed with real-world noise. This was an expected result, as the model was initially designed without taking real-world noisy data into consideration.

7 Conclusion

From our research study we found that both transportation, security, and social networking all have received a lot of attention for their use of trajectory prediction. Our major goal with this thesis was to analyse and predict trajectories inside indoor spaces, utilising historical data and contextual information that was automatically obtained by a positioning system.

Rather than using typical machine learning approaches which dominate the field of trajectory prediction, we relied on simple and easily recognised patterns. While machine learning provides numerous solutions, it often requires a lot of training data and can be computationally expensive. Our heuristic model which we discussed in our solution chapter is straightforward, easy to understand, and does not require large training datasets that have to be retrained whenever new data is available. As a result, it is suitable for making accurate predictions based on observed behaviours in controlled contexts. The indoor space is fixed, external factors like the weather or traffic are not present, and people's movements often revolve around habits or routines. Such environments are a good setting for making accurate predictions based on observed behaviours.

A central aspect of our research was the use of the spatio-temporal database system MobilityDB. We focused on comprehending movement patterns and drawing meaningful insights from them, rather than being entangled by the complexities and technicalities of the spatio-temporal data. MobilityDB provides significant advantages when used to manage spatio-temporal data. It speeds up the process of retrieving trajectory data, which consists of both a spatial and temporal element. More importantly, MobilityDB's organised storing and querying techniques simplified the analysis of temporal habits, which was a key component of our research.

8 Future Work

As we move forward, there are a few more areas we think can push our research even further. One of the main challenges is extending our model from indoor to outdoor trajectory predictions. Indoors, our approach takes advantage of rather constant patterns. Considering how people travel around a building, hallways, rooms and communal areas are all connected to each other via fixed routes. People frequently take the same route from their office entrance to the lab or restroom. Our model is intended to learn from these recurring actions, creating predictions based on historical data and the constraints of the geospatial layout.

Once outdoors, the number of possible routes and variations increases significantly. Instead of a fixed passageway to guide travel, there are streets, parks, shortcuts, and plenty of other options. The environment is more dynamic, with factors such as weather, traffic, and special events impacting how one chooses to go from point A to point B.

Another challenge we want to tackle is the use of real-world data. We've seen good results with our simulated data, but real-world setting combined with the inherent noise is always more unpredictable.

On the performance side, there's a feature within MobilityDB, namely spatio-temporal boxes. These have been mentioned before in Section 4.4.1, but have not been used yet. Improving the query processing will result in more efficient and faster predictions, which is essential for real-time trajectory prediction.

In conclusion, our research has laid a fundamental foundation for exploring the trajectory prediction field without resorting to machine learning. However, there are still various challenges and opportunities that can be addressed in future studies.

References

[1] Chujie Wang et al. “Exploring Trajectory Prediction Through Machine Learning Methods”. In: *IEEE Access* PP (July 2019), pp. 1–1. DOI: 10.1109/ACCESS.2019.2929430.

[2] Samet Ayhan and Hanan Samet. “Aircraft Trajectory Prediction Made Easy with Predictive Analytics”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* May 2016, pp. 21–30. DOI: 10.1145/2939672.2939694. URL: https://dl.acm.org/doi/10.1145/2939672.2939694.

[3] Shangbo Mao et al. “An Automatic Identification System (AIS) Database for Maritime Trajectory Prediction and Data Mining”. In: *Advances in Spatial Data Handling.* May 2018, pp. 241–257. ISBN: 978-3-319-57420-2. DOI: 10.1007/978-3-319-57421-9_20.

[4] Meng Chen, Yang Liu, and Xiaohui Yu. “NLPMM: A Next Location Predictor with Markov Modeling”. In: *Advances in Natural Language Processing.* May 2014, pp. 186–197. ISBN: 978-3-319-06604-2. DOI: 10.1007/978-3-319-06605-9_16.

[5] Amin Sadri et al. “What Will You Do for the Rest of the Day?: An Approach to Continuous Trajectory Prediction”. In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2 (Dec. 2018), pp. 1–26. DOI: 10.1145/3287064.

[6] Yanjun Huang et al. “A Survey on Trajectory-Prediction Methods for Autonomous Driving”. In: *IEEE Transactions on Intelligent Vehicles* 7 (Sept. 2022), pp. 1–1. DOI: 10.1109/TIV.2022.3167103.

[7] Jianwu Fang et al. *Heterogeneous Trajectory Forecasting via Risk and Scene Graph Learning.* Nov. 2022. DOI: 10.48550/arXiv.2211.00848.

[8] Rainu Nandal. “Spatio-Temporal Database and Its Models: A Review”. In: *IOSR Journal of Computer Engineering* 11 (Jan. 2013), pp. 91–100. DOI: 10.9790/0661-11291100.

[9] Deepika Shukla et al. “Comparing Oracle Spatial and Postgres PostGIS”. In: *Semantic Scholar.* 2016. DOI: 10.1007/978-3-319-57421-9_20. URL: https://api.semanticscholar.org/CorpusID:212479447.

[10] Jaymin Patel. “Temporal Database System”. In: *Department of Computing, Imperial College, University of London* 18 (2003).

[11] Richard Snodgrass and llsoo Ahn. “Temporal Databases”. In: *Computer* 19 (Oct. 1986), pp. 35–42. DOI: 10.1109/MC.1986.1663327.

[12] Md Mahbub Alam, Luís Torgo, and Albert Bifet. “A Survey on Spatio-temporal Data Analytics Systems”. In: *ACM Computing Surveys* 54 (Jan. 2022). DOI: 10.1145/3507904.

[13] Richard T Snodgrass. “Temporal databases”. In: *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space: International Conference GIS—From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning Pisa, Italy, September 21–23, 1992 Proceedings.* Springer. 2005, pp. 22–64.

[14] Christian S. Jensen and Richard T. Snodgrass. “Time Interval”. In: *Encyclopedia of Database Systems.* Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, 2009, pp. 3112–3113. ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_1423. URL: https://doi.org/10.1007/978-0-387-39940-9_1423.

[15] Donna Peuquet. “It’s About Time: A Conceptual Framework for the Representation of Temporal Dynamics in Geographic Information Systems”. In: *Annals of the Association of American Geographers - ANN ASSN AMER GEOGR* 84 (Sept. 1994), pp. 441–461. DOI: 10.1111/j.1467-8306.1994.tb01869.x.

[16] Esteban Zimányi et al. “MobilityDB: A Mainstream Moving Object Database System”. In: *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*. SSTD ’19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 206–209. ISBN: 9781450362801. DOI: 10.1145/3340964.3340991. URL: https://doi.org/10.1145/3340964.3340991.

[17] Esteban Zimányi, Mahmoud Sakr, and Arthur Lesuisse. “MobilityDB: A Mobility Database Based on PostgreSQL and PostGIS”. In: *ACM Trans. Database Syst.* 45.4 (Dec. 2020). ISSN: 0362-5915. DOI: 10.1145/3406534. URL: https://doi.org/10.1145/3406534.

[18] Hannah Bast et al. “Route Planning in Transportation Networks”. In: *Lecture Notes in Computer Science*. Vol. 9220. Nov. 2016, pp. 19–80. ISBN: 978-3-319-49486-9. DOI: 10.1007/978-3-319-49487-6_2.

[19] Xinmin Tang, Ping Chen, and Yu Zhang. “4D Trajectory Estimation Based on Nominal Flight Profile Extraction and Airway Meteorological Forecast Revision”. In: *Aerospace Science and Technology* 45 (June 2015). DOI: 10.1016/j.ast.2015.06.001.

[20] Meng Chen, Xiaohui Yu, and Yang Liu. “Mining Moving Patterns for Predicting Next Location”. In: *Information Systems* 54 (July 2015). DOI: 10.1016/j.is.2015.07.001.

[21] Julian Kooij et al. “Context-Based Pedestrian Path Prediction”. In: *Advances in Spatial Data Handling*. May 2014, pp. 241–257. ISBN: 978-3-319-57420-2. DOI: 10.1007/978-3-319-57421-9_20.

[22] Mikolaj Morzy. “Mining Frequent Trajectories of Moving Objects for Location Prediction”. In: *Machine Learning and Data Mining in Pattern Recognition*. July 2007, pp. 667–680. ISBN: 978-3-540-73498-7. DOI: 10.1007/978-3-540-73499-4_50.

[23] Hoyoung Jeung et al. “A Hybrid Prediction Model for Moving Objects”. In: *Proceedings - International Conference on Data Engineering*. May 2008, pp. 70–79. ISBN: 978-1-4244-1836-7. DOI: 10.1109/ICDE.2008.4497415.

[24] Shaojie Qiao et al. “Predicting Long-Term Trajectories of Connected Vehicles via the Prefix-Projection Technique”. In: *IEEE Transactions on Intelligent Transportation Systems* PP (Oct. 2017), pp. 1–11. DOI: 10.1109/TITS.2017.2750075.

[25] Anthony J.T. Lee. “Mining frequent trajectory patterns in spatial–temporal databases”. In: *Information Sciences* 179.13 (2009). Special Section on High Order Fuzzy Sets, pp. 2218–2231. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2009.02.016.

[26] Prajakta S Kalekar et al. “Time series forecasting using holt-winters exponential smoothing”. In: *Kanwal Rekhi school of information Technology* (2004), pp. 1–13.

[27] David Liben-Nowell and Jon Kleinberg. “The Link Prediction Problem for Social Networks”. In: CIKM ’03. New York, NY, USA: Association for Computing Machinery, 2003, pp. 556–559. ISBN: 1581137230. DOI: 10.1145/956863.956972. URL: https://doi.org/10.1145/956863.956972.

[28] Aswathy Divakaran and Anuraj Mohan. “Temporal Link Prediction: A Survey”. In: *New Generation Computing* 38 (Aug. 2019), pp. 1–46. DOI: 10.1007/s00354-019-00065-z.

[29] Chunsheng Fang et al. “Graph Embedding Framework for Link Prediction and Vertex Behavior Modeling in Temporal Social Networks”. In: *Proceedings of the ACM SIGKDD Workshop on Social Network Mining and Analysis*. Aug. 2011.

[30] Daniel Ashbrook and Thad Starner. “Starner, T.: Using GPS to Learn Significant Locations and Predict Movement across multiple users. Personal and Ubiquitous Computing 7(5), 275-286”. In: *Personal and Ubiquitous Computing* 7 (Oct. 2003), pp. 275–286. DOI: 10.1007/s00779-003-0240-0.

[31] Xiaoyang Liu et al. “Link Prediction Approach Combined Graph Neural Network with Capsule Network”. In: *Expert Systems with Applications* 212 (2023), p. 118737. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.118737. URL: https://www.sciencedirect.com/science/article/pii/S0957417422017559.

[32] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-term Memory”. In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[33] Alexandre Alahi et al. “Social LSTM: Human Trajectory Prediction in Crowded Spaces”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 961–971. DOI: 10.1109/CVPR.2016.110.

[34] Hao Xue, Du Q. Huynh, and Mark Reynolds. “Bi-Prediction: Pedestrian Trajectory Prediction Based on Bidirectional LSTM Classification”. In: *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. 2017, pp. 1–8. DOI: 10.1109/DICTA.2017.8227412.

[35] Yingfan Huang et al. “STGAT: Modeling Spatial-Temporal Interactions for Human Trajectory Prediction”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 6271–6280. DOI: 10.1109/ICCV.2019.00637.

[36] Simin S. Mirvahabi and Rahim Abbaspour. “AUTOMATIC EXTRACTION of IndoorGML CORE MODEL from OpenStreetMap”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XL-1-W5. Nov. 2015. DOI: 10.5194/isprsarchives-XL-1-W5-459-2015.

[37] Wen Liu et al. “Survey on CSI-based Indoor Positioning Systems and Recent Advances”. In: *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2019, pp. 1–8. DOI: 10.1109/IPIN.2019.8911774.