# Let's Speak Trajectories: A Vision to Use NLP Models for Trajectory Analysis Tasks

MASHAAL MUSLEH, University of Minnesota Twin Cities, Minneapolis, United States
MOHAMED F. MOKBEL, University of Minnesota, Minneapolis, United States

The availability of trajectory data combined with various real-life practical applications has sparked the interest of the research community to design a plethora of algorithms for various trajectory analysis techniques. However, there is an apparent lack of full-fledged systems that provide the infrastructure support for trajectory analysis techniques, which hinders the applicability of most of the designed algorithms. Inspired by the tremendous success of the Bidirectional Encoder Representations from Transformers (BERT) deep learning model in solving various Natural Language Processing tasks, our vision is to have a BERT-like system for trajectory analysis tasks. We envision that in a few years, we will have such system where no one needs to worry again about each specific trajectory analysis operation. Whether it is trajectory imputation, similarity, clustering, or whatever, it would be one system that researchers, developers, and practitioners can deploy to get high accuracy for their trajectory operations. Our vision stands on a solid ground that trajectories in a space are highly analogous to statements in a language. We outline the challenges and the road to our vision. Exploratory results confirm the promise and possibility of our vision.

CCS Concepts: • **Information systems** → **Spatial-temporal systems;**

Additional Key Words and Phrases: Trajectory analysis, trajectory NLP, trajectory operations

**ACM Reference Format:**
Mashaal Musleh and Mohamed F. Mokbel. 2024. Let's Speak Trajectories: A Vision to Use NLP Models for Trajectory Analysis Tasks. *ACM Trans. Spatial Algorithms Syst.* 10, 2, Article 15 (June 2024), 25 pages. https://doi.org/10.1145/3656470

## 1 INTRODUCTION

Enabled by location tracking technologies, there has been a vast amount of trajectories collected by industry and academia. Many of such datasets have been publicly released for various cities around the world, including Athens [5], Beijing [138], New York [94], Porto [99], Rio de Janeiro [30], Rome [7], San Francisco [1], Shenzhen [122], and Singapore [45]. This has enabled a myriad of trajectory analysis techniques that have been the focus of the spatial community for years (e.g., see References [113, 126, 151] for surveys). Such techniques have empowered numerous applications with high impact across many sectors. For example, in transportation, trajectory

This work is supported by the National Science Foundation (NSF), USA, under Grant IIS-2203553.
Author's addresses: M. Musleh, Computer Science and Engineering, University of Minnesota Twin Cities, 4-192 KHKH Building 200 Union Street SE, Minneapolis, MN, 55455, USA; e-mail: musle005@umn.edu; M. F. Mokbel, Computer Science and Engineering, University of Minnesota, 4-192 KHKH Building 200 Union Street SE, Minneapolis, MN, 55455, USA; e-mail: mokbel@umn.edu.

ACM 2374-0353/2024/06-ART15
https://doi.org/10.1145/3656470

analysis has been vital in data-driven routing [95, 144], traffic monitoring [52, 81], and traffic forecasting [43, 119]. In location-based services, trajectory analysis was used in trip planning [9, 154], route recommendations [16, 36], and map services [87]. In urban planning, trajectory analysis has been a key to map inference [39, 111], deciding on the locations of bike lanes and EV charging stations [40, 63] and understanding human mobility [67, 133]. In the health domain, trajectory analysis has played an important role in contact tracing [3, 84, 132] and understanding the pandemic spread [13, 114].

All such applications have to tackle a wide range of trajectory problems, including (a) trajectory similarity search, where the objective is to find those trajectories that are considered similar to each other according to some predefined similarity measure [15, 27, 61, 68, 98, 136, 143]; (b) trajectory imputation, where the objective is to add artificial points to a trajectory as a means of filling in the gaps between actual trajectory points [58, 62, 73, 125, 130, 150]; (c) trajectory classification and clustering, where the objective is to cluster or classify trajectories either based on their modality, similarity, location, or other characteristics [86, 104, 112, 121, 152, 153]; (d) trajectory prediction, where the objective is to predict the next few locations of the current trajectory points [33, 47, 69, 124, 142]; and (e) trajectory simplification, where the objective is to sample some of the trajectory points without losing its main characteristics [48, 54, 71, 72, 128, 148].

However, with all of these techniques, there is an apparent lack of full-fledged systems that provide the infrastructure support for trajectory analysis tasks. Existing attempts to build such systems (e.g., see References [2, 23, 55, 108]) are limited to providing the underlying index and retrieval storage but did not reach to the stage for providing complete data analysis functionality. The main reason is that the focus of trajectory analysis techniques is mainly on the algorithmic part of the analysis, which gives much less weight to the need of having a solid system infrastructure. Hence, despite the fact that all of trajectory analysis techniques deal with the same trajectory data, each of the proposed solutions is entirely designed to solve one problem of interest. This makes it hard and not practical to have a unified efficient system that is capable of supporting most (if not all) trajectory problems if each solution is entirely different.

Trying to learn from other communities, the research landscape of **Natural Language Processing (NLP)** was very recently in a similar situation. There have been decades of research in pushing the accuracy and efficiency of various NLP tasks, e.g., text similarity, text classification, sentence completion, and sentiment analysis. This has led to a myriad of different solutions for each of these problems, even though all tasks are for the same textual data. This is mainly for the same reasons that trajectory analysis techniques are entirely different from each other. Yet, most recently, in 2018, the **Bidirectional Encoder Representations from Transformers (BERT)** deep learning model [29] was proposed by Google to act as a unified solution infrastructure for a wide variety of NLP tasks. BERT, at its core, is equipped with the necessary NLP infrastructure to solve various NLP tasks, which only needs to be externally tuned with minimal overhead for each task. Examples of NLP tasks that used the BERT model include sentiment analysis [31], question answering [19], spell checking [146], text classification [22], text generation[17], text summarization [100], among others [59, 96]. BERT has also been used for similar problems with respect to speech processing, where the words are spoken instead of written [53, 117]. As a testimony to the importance and ubiquity of BERT to NLP research, the main BERT paper [29] has been cited more than 60K times within 5 years.

This article presents our vision toward using the same idea of BERT to magically deal with almost all trajectory analysis tasks. The goal is that BERT (or a customization of it) will do for trajectory analysis what it did already for NLP tasks. Should we be able to do so, various trajectory analysis ideas will be just about how to tune that BERT customized model to support the required analysis. Such a vision will lead to a long-waited-for full-fledged trajectory data management system that not only stores and indexes trajectory data but also natively supports all its analysis needs. Our vision is grounded by the fact that we can actually think of trajectories as statements. In a nutshell, a statement is composed of a set of words drawn from a set of limited words (language), while a trajectory is represented by a set of GPS points, which is also drawn from a set of possible points (space). Statements follow rules imposed by the underlying language, while trajectories follow rules imposed by both the underlying road network and the physical world. Words in a statement should be semantically related, while points in a trajectory should be spatially and temporally related.

While it is theoretically possible to think of trajectories as statements, and hence trajectories can be fed to BERT as is to support various trajectory analysis tasks, this may not practically work due to various inherent limitations in both the BERT model structure and the nature of trajectory data. Hence, our vision is composed of two orthogonal directions. The first direction is to overcome the limitations coming from the nature of trajectory data, where we can customize trajectory data to be fit for BERT and then use BERT as is. The second direction is to overcome the limitations of the BERT model itself and make it spatially and temporally aware in a way that it can support trajectory data. Both directions can be sought after together for the best performance. In all cases, the outcome of the vision is a BERT-like model that is not specific to one trajectory problem. Instead, it will act as a Swiss army knife that supports a myriad of diverse trajectory operations.

The rest of this article is organized as follows: Section 2 outlines the similarities between trajectories and statements, which presents the solid ground of our vision. Section 3 presents how BERT can be applied as is to five widely used trajectory analysis tasks, namely trajectory imputation, trajectory prediction, trajectory classification, trajectory simplification, and trajectory similarity. The challenges that face our vision, which emerge from our attempts to use BERT as is for various trajectory analysis tasks, are outlined in Section 4. Section 5 presents our vision with its two orthogonal but complementary directions to use BERT for trajectories. Initial exploratory experimental results that show the promise of our vision are presented in Section 6. Section 7 presents the related work to our vision. Finally, Section 8 concludes the article.

## 2 POINTS IN TRAJECTORIES VS WORDS IN STATEMENTS

This section presents the rationale behind our vision, where we see that points in trajectories follow very similar properties to words in statements. Hence, tools and techniques that are being used in NLP tasks (e.g., BERT) can be employed to support trajectory analysis tasks. We outline four such common properties, namely, *limited domain*, *domain constraints*, *intra-relationship constraints*, and *clear context*.

### 2.1 Limited Domain

Both words in a statement and points in trajectory are drawn from a limited domain defined by the underlying language or space, respectively. In particular, a *statement* is composed of an ordered set of *words* drawn from a finite pool of *words* (domain) per the underlying *language*. Similarly, a *trajectory* is composed of an ordered set of *points* drawn from a finite pool of *points* (domain) per the underlying *space*. This particular property is key to BERT functionality when dealing with statements. In particular, when using BERT for a certain language, it is first fed with large numbers of statements from that language as training examples, which could be coming from any set of online documents, e.g., Wikipedia articles. Then, BERT uses these documents to learn the domain of all possible words in the given language. That domain is then used to control BERT internal operations allowing it to understand any given set of statements and hence perform NLP operations over it. Our vision is based on the fact that trajectories exhibit the same limited domain property as statements. Hence, one can feed BERT with large number of trajectories in a certain city instead of statements in a certain language. Then, BERT can use these trajectories to learn the domain of all possible points in the city and use that knowledge to understand any given set of trajectories and perform various analysis tasks on it.

## 2.2 Domain Constraints

Although statements and trajectories must pick their words and points from their corresponding domains, the order of such words and points must adhere to some domain constraints. In particular, *words* in a *statement* must adhere to the constraints imposed by the underlying *language grammar*. Similarly, *points* in a trajectory must adhere to the constraints imposed by the underlying *road network and physical space*. BERT is taking advantage of this to guide its operations. For example, when fed by large number of documents, BERT would understand the grammar by knowing that the verb "are" is only used with plural nouns. Hence, when given a new statement that does not follow this constraint, BERT may raise a flag of grammatical error. Also, BERT can use this knowledge to predict the next word. Our vision is based on the analogy that along the same lines, when fed by large number of trajectories, BERT can understand several constraints. Examples include understanding that the speed of movement never exceeds a certain limit or that the change of speed from one trajectory segment to another is within a certain range. Even further, it can infer parts of the underlying road network from the large set of trajectory GPS points it has. Hence, when receiving a new set of trajectories, BERT can validate it against its learned constraints and flag for errors if any. Also, it can predict the next point or impute missing points accordingly without invalidating the trajectory domain constraints.

## 2.3 Intra-Relationship Constraints

Both words in a statement and points in a trajectory must be intra-related. In particular, *words* in a *statement* are *semantically* related, where random *words* cannot make a *statement*. Similarly, *points* in a *trajectory* are *spatially and temporally* related, where random *points* cannot make a *trajectory*. For example, the statement "John is drinking steak" satisfies the domain constraints (i.e., correct grammar); however, it does not satisfy the intra-relationship constraint, where "drinking" is not semantically related to "steak." BERT takes advantage of such constraints as it learns them from its input documents. Hence, when used to check typos, find missing words or predict the next words, BERT will use its learnt intra-relationship constraint and suggest the use of the word "eating" instead of "drinking," as it is more semantically related to the word "steak." Our vision is based on the analogy of the intra-relationship constraint to the case of trajectories. A sequence of GPS points that still match the domain constraints (i.e., road network and physical) may not satisfy the intra-relationship constraint, where one point of such sequence could be either from a nearby road or from the same road but in a different direction. When fed by trajectories, BERT should be able to learn such constraints and use them for various trajectory analysis tasks.

## 2.4 Clear Context

There is always a clear context that impacts the sequence of words in a statement or points in a trajectory. In particular, the *words* used in a *statement* would differ based on the *topic of discussion* (context). Similarly, the *points* used in a *trajectory* would differ based on the *driving modality* (context). For example, the words used in a medical document are pretty different than those words used in a political document, even though both documents are in the same language. BERT uses this property to learn the underlying context of statements in a document and uses this knowledge to perform various NLP tasks. If BERT can identify that a certain statement or document is coming from a medical context, then it will act differently (in terms of looking at different vocabulary) from the case where the statement is coming from a political context. With a similar analogy, our vision is based on the fact that trajectories also follow a clear context. For example, the driving modality (e.g., vehicles, buses, motorbikes, bikes) impact the sequence of points used in a trajectory in terms of different speeds, different driving patterns, and even different lanes and roads. Hence, when fed by trajectories, BERT could understand the underlying context of these trajectories and then use this knowledge to understand the context of any new given trajectory and perform various tasks on it. For example, if BERT can identify that a certain trajectory belongs to a bike, then it will act differently from the case if the trajectory is coming from a bus.

## 3 BERT FOR TRAJECTORY ANALYSIS TASKS

This section discusses the first step toward our vision by showing that five different widely used trajectory analysis tasks are pretty analogous to corresponding five widely NLP analysis tasks. In particular, we show that *trajectory imputation* is analogous to *find the missing word* problem (Section 3.1), *trajectory prediction* is analogous to *next sentence prediction* (Section 3.2), *trajectory classification* is analogous to *text classification* (Section 3.3), *trajectory simplification* is analogous to *text summarization* (Section 3.4), and *trajectory similarity* is analogous to *text similarity* (Section 3.5). Since BERT is widely used to solve all these five NLP tasks, with the same analogy we also show that BERT can *potentially* be used to solve their corresponding five trajectory analysis tasks. We use the word *potentially* here as this may not practically work right away. This is due to many challenges that we will outline later in Section 4 and need to address to realize our vision of a BERT-like model to execute various trajectory analysis tasks.

### 3.1 BERT For Trajectory Imputation

This section presents the analogy between the "*trajectory imputation*" problem and "*Finding the missing word*" NLP task, which is commonly solved using BERT.

**Trajectory Task: Imputation.** Trajectory data are inherently sparse, with large and frequent spatial and temporal gaps between every two consecutive GPS readings. This is mainly either due to low GPS sampling rate to preserve the bandwidth, battery, and/or storage in location tracking devices or due to loss of GPS signal in some areas such as tunnels and near high-rise buildings. Such gaps present an inherent uncertainty of the object's whereabouts between each two GPS readings, which affects all applications that rely on trajectory data. The higher the sparsity (i.e., the larger and more frequent such gaps spatially and temporally), the lower the accuracy and quality of both trajectory data and the applications that rely on it. To address the sparsity issue, and as a means of boosting the accuracy of trajectories and their applications, several recent efforts were dedicated to insert artificial location points between each two consecutive trajectory points. The promise is that these artificially imposed points are as accurate as if they were obtained by actual GPS readings of trajectory data. Such a process had various names, including trajectory interpolation [73, 150], trajectory completion [62], trajectory data cleaning [145], trajectory restoration [58], trajectory map matching [8, 74], trajectory recovery [125, 130], and trajectory imputation [14]. Without loss of generality, we will use the term "trajectory imputation" in this article. With the exception of few techniques [32, 62, 88], the large majority of existing trajectory imputation techniques rely on matching the trajectories on the underlying road network, and hence they have an implicit assumption that the underlying road network is available and reliable, which is not always true. Road networks, like any other data, suffer from all sorts of inaccuracy and may not be even available in many places [79, 82, 91, 116]. This calls for developing new imputation techniques that do not require the knowledge of the underlying road networks.

**NLP Task: Finding the Missing Word.** Consider, for example, the incomplete English statement "Paris, the ... of France, is ... Summer Olympics in 2024," where each blank "..." represents a missing

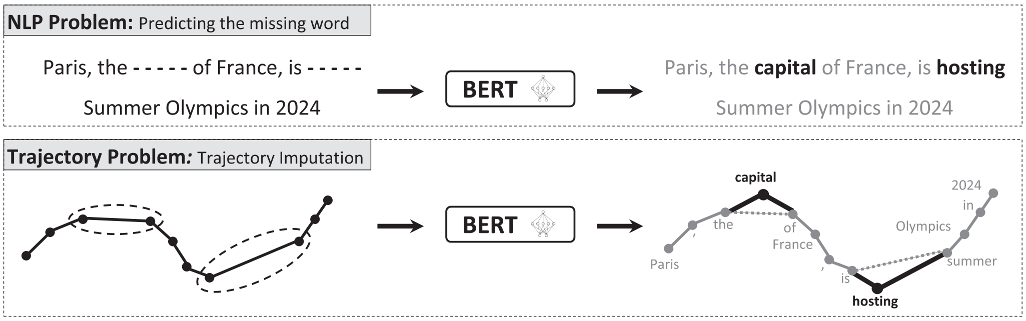

NLP Problem: Predicting the missing word
Paris, the ----- of France, is -----
Summer Olympics in 2024
BERT
Paris, the capital of France, is hosting
Summer Olympics in 2024
Trajectory Problem: Trajectory Imputation
BERT
Paris
the
capital
of
France
is
hosting
summer
Olympics
in
2024

Fig. 1. Trajectory imputation using BERT model.

word. *Finding the missing word problem* (a.k.a cloze test) aims to infer the words that are replaced by a blank. In this case, the first missing word would be "capital," and the second missing word is "hosting." This is one of the main and very common tasks in NLP, as there are several practical scenarios behind having missing words in a statement. For example, speech and image recognition techniques may partially recognize a statement, with some (missing) words that could not be recognized and left as blank. A translation task may miss translating some words and leave them as blank or mark them as low confidence. A typo in a text could be replaced by a blank if it was not corrected by a spell checker. In such scenarios, finding the missing word problem aims to fill in the blanks. BERT has been used to solve finding the missing world problem, where it is first trained by hundreds of thousands of true statements that will make it able to understand the context of any sentence and accurately find out its missing words.

**The Analogy.** Figure 1 shows the analogy between *trajectory imputation* and *finding the missing word* problems. Instead of the blanks "...," we circle the segments that have significant gaps between their end points. For clarity, the example aligns the statement and trajectory together where the statement is composed of 11 words (including two punctuation marks) with 2 more words marked as blank, while the trajectory is composed of 11 points with two segments identified as need to be imputed. The question that we are asking in our vision here is that if BERT, when trained with large number of statements, can identify the two missing words as "Capital" and "Hosting," then can BERT be trained with large number of trajectories and then used to impute a trajectory by identifying its missing points? Should we be able to do so, we would be solving the trajectory imputation problem without the need for the knowledge of the underlying road network, making the solution applicable to much wider set of problem settings.

## 3.2 BERT For Trajectory Prediction

This section presents the analogy between the "*trajectory prediction*" problem and "*next sentence prediction*" NLP task, which is commonly solved using BERT.

**Trajectory Task: Prediction.** Trajectory prediction is the task of predicting the next few points of a current trajectory. This is one of the very common tasks in trajectory analysis as it is a cornerstone to many practical applications. For example, knowing where current vehicle trajectories are heading to enables traffic monitoring and forecasting where congestions can be expected ahead of time, and hence an appropriate action can be taken [46, 50]. It also enables events forecasting where one can predict expected gathering events at certain locations [49, 119]. Wireless communications can strengthen cellular connectivity by preparing the next few predicted cell towers to admit mobile devices based on the predicted workload [25, 93, 110]. Personalized services and recommendations, which include offering location-based information or advertisements, benefit

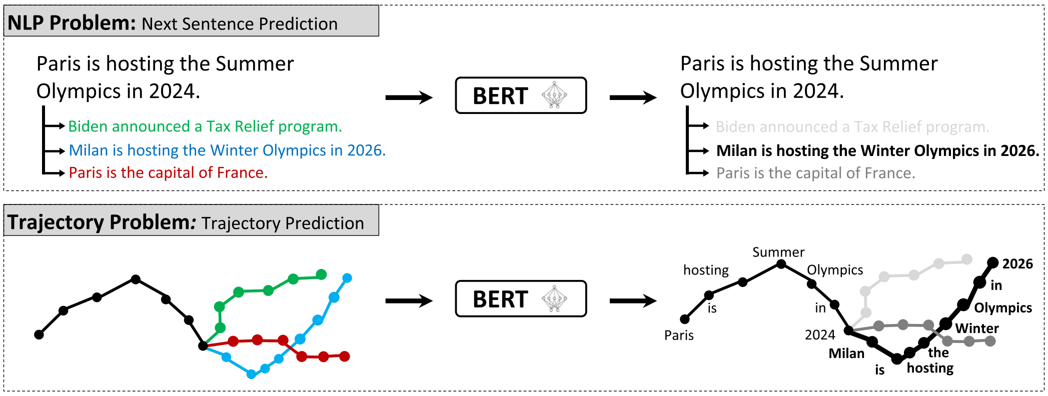

NLP Problem: Next Sentence Prediction
Paris is hosting the Summer Olympics in 2024.
Biden announced a Tax Relief program.
Milan is hosting the Winter Olympics in 2026.
Paris is the capital of France.
BERT
Paris is hosting the Summer Olympics in 2024.
Biden announced a Tax Relief program.
Milan is hosting the Winter Olympics in 2026.
Paris is the capital of France.
Trajectory Problem: Trajectory Prediction
BERT
Paris
is
hosting
Summer
Olympics
in
2024
Milan
is
hosting
the
Winter
Olympics
in
2026

Fig. 2. Trajectory prediction using BERT model.

from predicting vehicle trajectories to personalize their offering [24, 115]. Overall, the ability to do trajectory prediction has enabled a whole area of research in spatio-temporal predictive query processing, where the objective is to support spatio-temporal queries asking about a future time rather than the current state [41, 42]. Due to its importance, significant efforts have been dedicated to support various forms of trajectory prediction including short-term prediction (next few minutes) [33], long-term prediction (next 20-30 minutes) [47, 142], or prediction for extended periods (e.g., the rest of the day) [105]. These solutions are based on several frameworks, including statistical patterns [38, 85], machine and deep learning [33, 47, 69], and Markov models [4, 80].

**NLP Task: Next Sentence Prediction.** Consider, for example, the English statement "Paris is hosting the Summer Olympics in 2024," the *next sentence prediction* aims to predict, with probability, what would be the next sentence among a set of options. For example, if the following three statements are potential ones: "Biden announced a Tax Relief program," "Milan is hosting Winter Olympics in 2026," and "Paris is the capital of France," then a *next statement prediction* algorithm may give probabilities 1%, 70%, and 29% for these statements, respectively, which recommends that the second statement is the most likely one to come next after the input statement. This has practical applications including sentence auto completion and text generation. The *next sentence prediction* problem is usually solved using a BERT model. To do so, BERT is first trained and fine-tuned using pairs of <*input*, *target*> sentences. It gets such pairs from its input training data, composed of hundreds of thousands of statements in documents. This will make BERT understand a target statement, given an input one, and hence is able to accurately predict that next target statement.

**The Analogy.** Figure 2 shows the analogy between *trajectory prediction* and *next statement prediction* problems. For the latter one, we plot the three candidate next statements in three different colors. For the case of trajectory prediction, we plot the three potential trajectories in the same three colors to their corresponding statements. For clarity, each (potential) trajectory has the same number of points as the number of words of its corresponding statement. Hence, one way to solve the trajectory prediction problem is to train a BERT model with real trajectories. Then, BERT can split these trajectories into large numbers of pairs of sequence trajectories in the form of *input* and *target* trajectories. In a way analogous to what BERT is doing for the *next sentence prediction*, it can use its trajectory-based trained model to predict (with probability) the next trajectory among the three possible options. Similarly to the case of trajectory imputation, a great advantage of using BERT for trajectory prediction is that it can do so without the need to know the underlying road network, which would distinguish it from all existing trajectory prediction approaches.

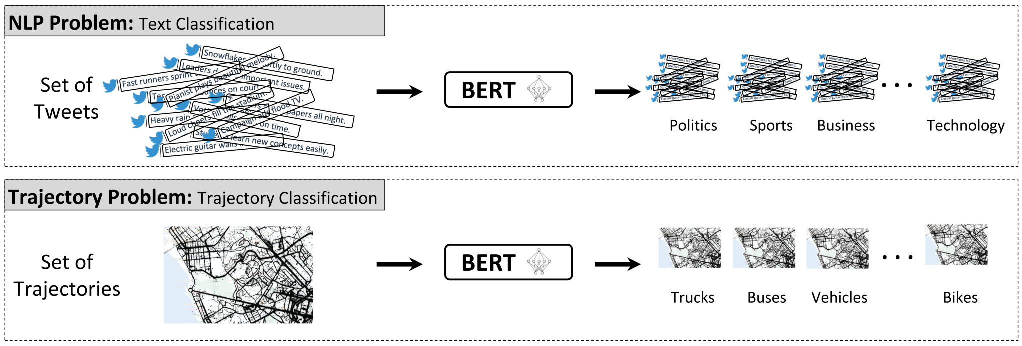


Fig. 3. Trajectory classification using BERT model.

## 3.3 BERT For Trajectory Classification

This section presents the analogy between the "*trajectory classification*" problem and "*text classification*" NLP task, which is commonly solved using BERT.

**Trajectory Task: Classification.** Trajectory classification is the process of associating a trajectory with one class from a predefined set of classes. A prime example of such an operation is associating a trajectory with a driving modality that could be bike, bus, vehicle, or even a walking trajectory [86, 104, 112, 121, 152, 153]. Such classification is crucial in many practical applications. For example, to infer an accurate travel time estimation (ETA) from a large set of trajectories, a pre-processing step needs to separate vehicle, bus, and motorbike trajectories from each other. Then, we use each set of trajectories separately to have a modality-based ETA. Map inference algorithms that are used to infer the underlying vehicle, bus, and bike routing maps would need to first classify a given set of trajectories based on their modality and use them separately to discover the corresponding map. Another example of trajectory classification is associating a trajectory with its travel purpose that could be work, commute, shopping, education, or recreation [6, 26, 75, 76, 109, 131]. This has a direct application in urban planning as a means of helping decision makers understand travel demands and relationships among neighborhoods. Existing approaches for trajectory classification rely on either hand-crafted rules for each class, heavily engineered features to train supervised machine learning models, or customized deep learning models.

**NLP Task: Classification.** Text classification is one of the most widely used analysis tasks in NLP for a large number of important and practical applications. The objective is that given a set of text, associate each text with one class from a predefined set of classes. A prime example would be classifying social media posts (e.g., tweets) into their topic category, e.g., sports, politics, news, technology, and health. This is an important preprocessing operation for various data analysis procedures that need to have the analysis based on only posts that belong to a certain category. Another example is sentiment analysis, where social media posts are classified per their sentiment category, e.g., happy, angry, sarcastic, and lukewarm. This is important in market study to understand the reaction of users to a certain product, advertisement, or article. The problem of text classification is usually solved using a BERT model. To do so, BERT is first trained on a large number of *unlabeled* sentences to learn about words in general and how they relate to each other. Then, it is fine-tuned using relatively smaller *labeled* sentences as <*tweet*, *category*>. Such a trained model is then used to decide on the category of a given tweet.

**The Analogy.** Figure 3 shows the analogy between *trajectory classification* and *text classification* problems, where text classification is used to classify tweets into politics, sports, business, and technology, while trajectory classification is used to classify trajectories into trucks, buses, vehicles, and bikes. Hence, one way to solve the trajectory classification problem is to go through a similar procedure of using BERT for the text classification problem. In particular, we start by training a BERT model using large numbers of *unlabeled* trajectories. Then, we fine-tune the model using a relatively smaller set of *labeled* trajectories of the form <*trajectory*, *modality*>. Finally, we use the refined model to infer the modality of a given trajectory.

## 3.4 BERT For Trajectory Simplification

This section presents the analogy between the "*trajectory simplification*" problem and "*text summarization*" NLP task, which is commonly solved using BERT.

**Trajectory Task: Simplification.** Trajectory simplification, sometimes seen as the opposite of trajectory imputation, is the task of reducing the number of trajectory GPS points while preserving their essential information. Trajectories can be vast and complex, making it costly and challenging to transmit (e.g., bandwidth in cellular phones is limited), store (e.g., managing and storing these large datasets can become prohibitively expensive), and process (e.g., executing queries on such datasets becomes more complex). Trajectory simplification lowers the number of points and thus significantly reduces the cost of query processing, storage, and data transmissions of complex trajectories. It can be employed either in online or offline mode. In online mode, during data collection, trajectory simplification helps tracking devices transmit only the most important points in real time. In offline mode, a simplified trajectory is obtained from the full trajectory and used either for storage or as a filter step during query processing, with refinements done using the full trajectory data. Due to the importance of trajectory simplification problem, significant research efforts were dedicated to solve it (e.g., see References [48, 54, 71, 72, 128, 148]). Most of these approaches aim to maintain a certain distance or direction error threshold when dropping some of the original points. Some of these approaches aim to match trajectories to the underlying road network and then simplify each trajectory by selecting representative road network points (e.g., intersections) [51, 60, 137].

**NLP Task: Text Summarization.** Consider, for example, the verbose English statement "Paris, the capital of France, is hosting the Summer Olympics in 2024." A *text summarization* procedure would make a shorter version of this statement to be: "Paris is hosting 2024 Olympics." Given a document of words, a text summarization analysis task aims to summarize the document by a short description that can be used for newsletters, video descriptions, or brief highlights. Such short description uses less number of words while preserving the main ideas of the original text and minimizing the information loss due to the removed text. The outcome summary can be either *extractive*, i.e., uses only words from the original text, or *abstractive*, i.e., can come up with entirely new words and phrases that were not present in the original text. In our example, we used an extractive summarization as all the words are taken from the original statement. BERT has been widely used for the text summarization problem. To do so, similarly to the case of text classification, BERT has to be first trained on large number of documents to build its initial model. Then, the model is fine-tuned using another smaller set of document and summary pairs to learn what would be the words to use or omit to come up with a document summary. Finally, given a document, BERT would use its learnt model to produce the summary.

**The Analogy.** Figure 4 shows the analogy between *trajectory simplification* and *text summarization* problems. For clarity, we plot the trajectory before and after simplification with the same number of points that corresponds to the statement before and after summarization. As in the case of text summarization where the summary still preserves the full meaning of the statement, the simplified trajectory still preserves the characteristics and overall structure of the full trajectory

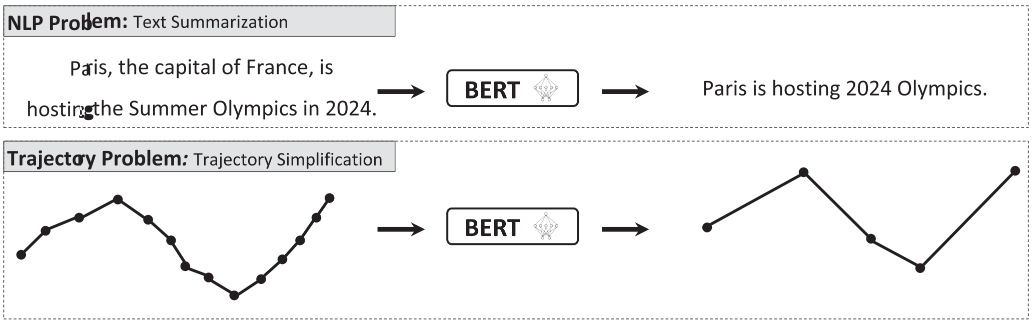

NLP Problem: Text Summarization
Paris, the capital of France, is hosting the Summer Olympics in 2024.
BERT
Paris is hosting 2024 Olympics.
Trajectory Problem: Trajectory Simplification
BERT

Fig. 4. Trajectory simplification using BERT model.

where main turns are still kept. As BERT is used for text summarization, one way to solve the trajectory simplification problem is to train a BERT model with large number of real trajectories and then fine-tune the model using pairs of the form <*raw trajectory*, *simplified trajectory*>. Though our example shows an *extractive* simplification, where the simplified trajectory uses only points from the full trajectory, the idea can also be applied to *abstractive* simplification. Similarly to previous trajectory analysis tasks, using BERT would allow trajectory simplification to be done without the need to be aware of the underlying road network.

## 3.5 BERT For Trajectory Similarity

This section presents the analogy between the "*trajectory similarity*" problem and "*text similarity*" NLP task, which is commonly solved using BERT.

**Trajectory Task: Similarity.** Trajectory similarity is the process of computing a similarity score between two trajectories based on their sampled GPS points. This is one of the commonly performed tasks in trajectory analysis, as it is a fundamental step to many practical applications. For instance, in routing and navigation applications, given a historical dataset of trajectories, a source point, and a destination point, it is often needed to compute the estimated travel times (ETA) for a certain route between source and destination points [87, 127]. Such information is highly valuable to help plan routes more efficiently or to add travel time information to the map. In this case, trajectory similarity is utilized to find trajectories that closely resemble the given route and then use these trajectories to help estimate the route travel time. In addition, several other trajectory analysis tasks rely on trajectory similarity. For example, in clustering and classification tasks (Section 3.3), computing the similarity score between two or more trajectories is essential. In particular, such tasks use the similarity scores to correctly assign each trajectory to an appropriate cluster based on its similarity to the other trajectories in each cluster. The higher the score to a certain cluster, the closer the trajectory to it and the higher the trajectory probability to belong to it. Another example of analysis tasks that utilize trajectory similarity is outlier detection. This is a very important preprocessing step for any downstream application that uses trajectories as it can help in cleaning such trajectories and ensure the application receives high-quality input data. In this task, given a dataset of trajectories, similarity scores are computed for each trajectory with respect to the reset of trajectories in the dataset. Then, those trajectories that are significantly less similar to the majority of the data are flagged as outliers, which then can be eliminated or processed independently via data cleaning tools. Due to the importance of trajectory similarity in various applications, significant research efforts have been devoted to it (e.g., see References [27, 61, 68, 136]). Many of these approaches rely on pairwise computations

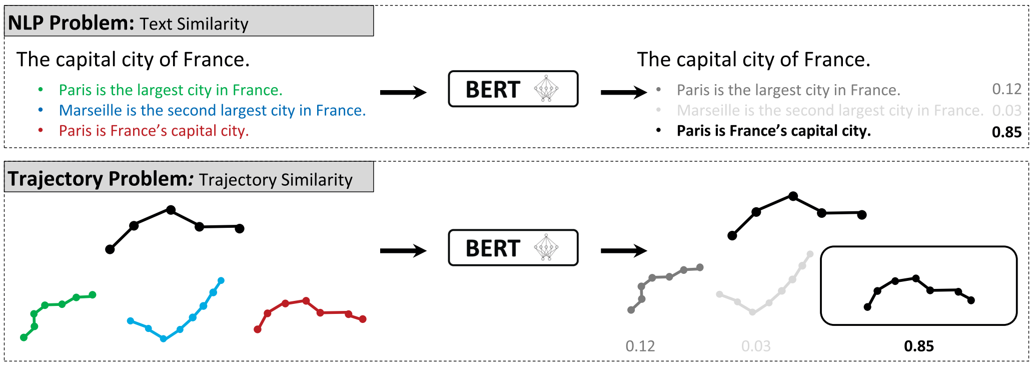

NLP Problem: Text Similarity
The capital city of France.
Paris is the largest city in France.
Marseille is the second largest city in France.
Paris is France's capital city.
BERT
The capital city of France.
Paris is the largest city in France. 0.12
Marseille is the second largest city in France. 0.03
Paris is France's capital city. 0.85
Trajectory Problem: Trajectory Similarity
BERT
0.12
0.03
0.85

Fig. 5. Trajectory similarity using BERT model.

between the sampled GPS points to find the similarity score, which may become prohibitively expensive with large datasets and large numbers of GPS points.

**NLP Task: Text Similarity.** Considering, for example, an input English statement such as "The capital city of France," the *text similarity* aims to compute how similar (or relevant) this statement is to a set of available statements or documents. For example, if the following three statements are the available ones: "Paris is the largest city in France," "Marseille is the second-largest city in France," and "Paris is France's capital city," then a *text similarity* algorithm may find the similarity scores between the input statement and the three available statements as 0.12, 0.03, and 0.85, respectively, which recommends that the last one is the most similar statement to the input. This has many practical applications, including search and information retrieval, results ranking, and question answering. This is in addition to clustering applications, e.g., categorizing a set of tweets into topics where tweets in each cluster are highly similar to each other. Similarly to many other NLP tasks, *text similarity* is usually solved using a BERT model. Given two statements, BERT represents them as two vectors of the same size, regardless of the number of words in each statement. The vectors then go through a mathematical similarity measure, e.g., cosine similarity or Euclidean distance, to measure their similarity. The similarity score is then used to represent how both statements are similar to each other. To do so, BERT is first trained on large datasets of statements so it can learn relationships between words and then be able to represent similar statements by similar vectors.

**The Analogy.** Figure 5 shows the analogy between *trajectory similarity* and *text similarity* problems. For the latter one, we have an input statement and three other available ones, which we plot in different colors. Similarly, for trajectory similarity, we have an input trajectory and three available ones. For clarity, each of the three trajectories has the same number of points as the number of words of its corresponding statement. We also plot the three trajectories in the same three colors to their corresponding statements. The goal is to compute their similarity to the input. Hence, one way to solve the trajectory similarity problem is to train BERT on a large trajectory dataset so it can learn how to represent similar trajectories by similar vectors. Then, we can use these vectors to easily compute the similarity between their corresponding trajectories. This has the potential to scale up, since the vector size is constant regardless of the number of GPS points, which would distinguish it from other existing pairwise similarity approaches.

## 4 CHALLENGES IN USING BERT FOR TRAJECTORY ANALYSIS

The clear analogy between trajectories and statements, and hence between their corresponding tasks, calls for applying state-of-the-art NLP models, e.g., BERT, for trajectory operations. The hope is that, given the apparent analogy, BERT will be as successful for trajectory analysis, as it is already successful for NLP tasks. However, this may not be practically applicable. In fact, if we apply BERT as is to trajectory analysis tasks, then it will yield highly inaccurate results and may not even work in most cases. This is mainly due to several fundamental differences between trajectories and statements, which hinder the applicability of BERT to trajectory operations. This section presents the following four main challenges that stem from these differences, which need to be addressed first to pave the way for our vision of a BERT-like model for a myriad of trajectory analysis tasks.

**Challenge I: Ratio of Training Datasets to Possible Words.** BERT is designed to be used for languages, where (a) the number of possible words is of limited size and (b) there is an abundance of publicly available data, including Wikipedia pages, news articles, and various websites. In fact, BERT was actually trained on ~3.3B word corpus (2.5B of them are from Wikipedia and 800M from Books Corpus [158]) composed of ~30K distinct words [29]. This means that, on average, each English word appears ~100K times in the BERT training set. This gives BERT the ability to see each word in various contexts and hence would be able to understand various context-based meanings of each word as well as linking words together when they co-occur often. This is basically due to the law of large numbers that BERT heavily relies on. Meanwhile, trajectory datasets exhibit a very different behavior. First, the number of possible points that can patriciate in a trajectory is huge and highly depends on the resolution of the GPS tracking devices. For example, assuming a 5-m accuracy of a GPS device, then a 1 km$^2$ area would have 40,000 GPS points. Considering a state like Minnesota, which has an area of around 225,000 km$^2$, we would have 9B possible GPS points in only one state. Of course, the numbers would be much less if we only focus on roads, biking trails, and sidewalks, which are the ones that appear in trajectory datasets, yet, still the numbers would be orders of magnitudes large than language datasets where the English word has only around 30K words (points). Second, due to many factors, including privacy and proprietary ownership, trajectory data are not widely publicly available like language data. With such limited data, distinct GPS points would hardly appear in training datasets. For example, a trajectory dataset from Oregon State, obtained from UCR STAR [118], has ~1.3M distinct GPS points with ~1.75M total points. This means that, on average, each trajectory point appears only once in the training datasets, which is five orders of magnitude *less* than what English words appear in their training datasets (100K times). Third, although trajectory datasets consist of a finite set of points, these trajectories (in their physical form on the ground) are in fact continuous signals with an infinite number of points. We only get a finite sequence of points for each trajectory, because these are the points that the GPS tracking device has sampled from the original continuous trajectory. This is contrary to language, where statements are actually discrete signals and do not have this sampling concept. This fundamental difference can translate into challenges in the mapping between NLP problems and trajectory analysis tasks. For example, in the trajectory imputation task, the number of missing points for a given trajectory segment can be (theoretically) infinite, while the number of missing words that BERT can solve has to be finite and predefined. Similarly, in the trajectory similarity task, consider two visually identical trajectories where one is more dense (i.e., has more sampled points on straight segments) than the other. Although we may expect BERT to represent them with similar vectors, BERT in reality may represent them completely differently, since the Transformer [120] (the core component of BERT) is very sensitive to the length of the sequence. Apparently, trajectory data characteristics are not suitable for BERT. In its core, BERT heavily relies on having each word appears hundreds of thousands of time in the training data, which would allow BERT to understand various contexts and witness different scenarios of each word. This makes BERT able to support various NLP tasks that all rely on the context understanding of BERT for each word in a statement. Hence, applying BERT as is to trajectory data, with its scarce data availability and large numbers of possible words, is not practical.

**Challenge II: Ratio of Noisy Data.** Language data are subject to noise that takes place in typos, grammatical errors, words that could not be extracted from images, or words that were not well translated from another language. Overall, noisy data are considered as outliers in the whole language dataset. BERT is kind of used to this, as given its large numbers of training datasets, it can identify the outlier noisy parts of the data and act accordingly. Meanwhile, noise in trajectory data is inherent, and it could be even more than accurate data. This is due to the inherent inaccuracy and the low sampling rate of GPS tracking devices as a means to accommodate bandwidth and battery limitations. Such high ratio of noisy data makes it very hard for BERT to learn from its own training set. Noisy data will increase the number of distinct points in a dataset, which will decrease the number of times that each point appears in the training set. Besides affecting the BERT training process, noisy data significantly impact the applicability of BERT to some trajectory analysis tasks. For example, consider the trajectory imputation task, which is analogous to finding the missing word NLP problem (Section 3.1). In the NLP problem, BERT is usually deployed to find only one missing word. Yet, errors that lead to trajectory imputation are usually coming from low sampling rates, which would drop out several points between each two consecutive points. Applying BERT as is to trajectory imputation will end up having only one point between the two segment end points. While this would increase the trajectory accuracy, it is not enough because we would need to fill in multiple points in between two segment end points.

**Challenge III: Long and Unrelated Consecutive Trajectories.** Statements are usually composed of few words, typically 15–20 words per statement. Then, paragraphs are composed of a set of *related* statements, typically five to eight statements per paragraph. Meanwhile, a trajectory may include hundreds of points, and subsequent trajectories may be *unrelated*, e.g., a series of taxi trips. BERT, by design, has a limit on the input length of each statement it can process, mainly due to its computationally expensive attention mechanism that exponentially grows with the number of words it needs to take into account. That limit for BERT is usually big enough to cover most language statements. In the case of long statements, BERT truncates them to keep only a window of the last $N$ words, where $N$ is the maximum input length. Apparently, the number of points in a trajectory is mostly above the limit that BERT can accommodate in its attention mechanism. Hence, it may not be practical for BERT to be applied as is for trajectories that are already long enough without splitting these trajectories into short ones and hence reduce their context information. Along the same lines, BERT takes full advantage of the fact that subsequent statements are related to each other. Hence, using its training set, BERT is able to understand the relations between subsequent statements and use it for various NLP tasks, including predicting the next statement task (Section 3.2). Since subsequent trajectories may not be that related, it would be harder for BERT to perform trajectory tasks that link a trajectory to its next one, e.g., trajectory prediction.

**Challenge IV: New Data and Changing Road Networks.** Once BERT is trained with language data, it can process any new statements, because they will be from the same underlying vocabulary (e.g., English) and follow the same language rules (e.g., grammar). However, if we train BERT with trajectory data of certain regions, then BERT may not be able to process new trajectories from a different region, because they use a completely new vocabulary that BERT has not seen before. Even if the new trajectories are from the same region, the road network could have changed since the last training, invalidating some rules that BERT may have learned. This makes using BERT as is with trajectories hard to scale to new regions or over time when the road network changes.

## 5 THE VISION: A BERT-LIKE MODEL FOR TRAJECTORY ANALYSIS

Our vision is that the spatial community would work together toward a full-fledged BERT-like system for a myriad of trajectory tasks. *We envision that, in a few years, we will have such system, where no one needs to worry again about each specific trajectory analysis operation. Whether it is trajectory imputation, similarly, or clustering, it would be one system that researchers, developers, and practitioners can deploy to get high accuracy for their tasks. The system would always be extensible in a way that can accommodate new operations contributed by the community at large.*[1]

This envisioned system does not have to be a completely new one built from scratch. Instead, it can be an adaptation of the current BERT system to make it more amenable to trajectory data analysis. Toward our vision, we believe that the community needs to explore two orthogonal, but complementary, directions. The first direction is to customize both the trajectories and the way we call BERT from the outside, while keeping the internal core architecture of BERT as is without any change. This will produce a quicker system solution that is much more accurate than using BERT without any customization as in Section 3 but still does not exploit the full power of BERT for more accurate results and system extensibility. The second direction is to inject spatial and temporal awareness inside the core of BERT itself. This will make BERT deal with spatial data in general and trajectories in particular as first-class citizens and support their special characteristics. As a result, this direction will produce a more accurate system as it exploits the full potential of BERT at its core. However, significant efforts would need to be put in to realize it. Exploiting and applying both directions together would produce a system with the ultimate desired performance for trajectory applications. Below are our initial thoughts on how to tackle each direction.

**Direction I: Customized Trajectories and BERT.** This direction aims to customize either the trajectories or the use of BERT or both together, without changing the BERT core, in a way that can potentially overcome some of the challenges listed in Section 4. Here are three examples of trajectory dataset customizations that belong to this direction: (1) Partition the space into a set of fine-grained hexagons, using Uber's H3 Hexagonal Hierarchical Spatial Index [10]. This way, all points within the same hexagon will be assigned to the same GPS value, which is the hexagon centroid. Then, trajectories become a sequence of hexagons instead of points, and the hexagons become the words to BERT. This customization brings the number of possible words/points in the example Oregon State dataset mentioned in Challenge I in Section 4 down from ~1.3M to ~18K, where now each point appears ~100 times on average during training. This is a two-order-of-magnitude improvement compared to the original dataset where each trajectory point appears only once in the training data. At the same time, trajectories in this form (i.e., sequence of tokens) become more of discrete signals rather than continuous ones, which is closer to the nature of statements. Such customization can potentially overcome both Challenge I and Challenge II, while still preserving accuracy due to the fine-grained nature of the hexagons. (2) Generate synthetic trajectories to enrich our corpus. To do so, we can employ existing trajectory simulation techniques (e.g., References [103, 149]), which basically take our available real trajectory data to generate additional trajectories that resemble the behavior of existing trajectories over different parts of the road network. The will enrich our corpus, which can be then used to train BERT, while avoiding more noisy data. Hence, this can potentially overcome both Challenge I and Challenge II. (3) Split long trajectories into a set of shorter subtrajectories. This will ensure that consecutive trajectories are both short and related, which would overcome Challenge III in Section 4. In other words, with this, trajectories would be actually analogous to paragraphs rather than statements. Recall that a *paragraph* is a collection of subsequent short and related *statements*. With splitting, a *trajectory* also becomes a collection of subsequent short and related *subtrajectories*.

[1]An earlier and shorter version of our vision is published in a four-page vision paper at ACM SIGSPATIAL 2022 [92].

Meanwhile, customizing the use of BERT may be task specific. We give an example for the trajectory imputation task (Section 3.1). Since BERT is designed to find only one missing word in a statement, it may not be suitable as is for trajectory imputation, where we would need to impute several points between each two GPS readings. A possible customization to overcome this issue is to call BERT iteratively. For example, in the second trajectory gap in Figure 1, we first call BERT to predict one missing point, which is the one shown in the figure corresponding to the word "hosting." Then, if we need an additional imputed point, we can call BERT again for the same gap by including the point/word that we just found ("hosting") in the input as if it was originally there. This is to get an additional imputed point for the same gap after the word "hosting." Once we get a new point, we can use it again in the input to call BERT for a third imputed point. We can iteratively repeat the process until reaching sufficient granularity. While this approach may be computationally expensive, it shows an example of customized usage that helps overcome Challenge II, which is related to GPS noise and low sampling rates. Another example of customizing the use of BERT is to partition the space into multiple regions and train a dedicated BERT model for each region using its trajectories. This can help overcome Challenge IV related to new data and changing road networks. In this case, we can identify regions with no or insufficient data, and then when more trajectory data in one of these regions become available later, we can trigger a training process for a new BERT model for this region. Similarly, this partitioning customization can also help overcome the issue of changing road networks. In this case, we can monitor the regions using tools that detect road network changes (e.g., RASED [90, 91]) and then trigger a re-training process for BERT models in regions where the roads have changed and new data have become available.

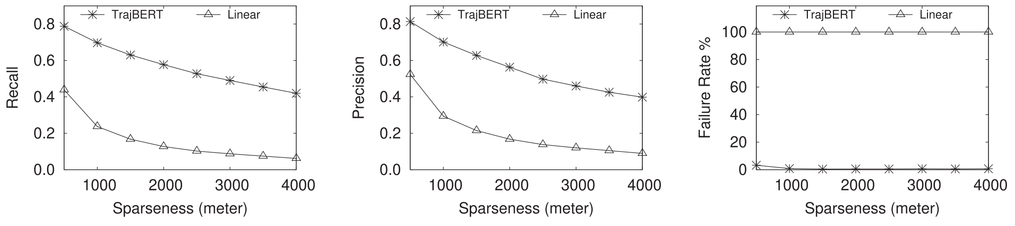
**Direction II: Spatially and Temporally Aware BERT.** This direction aims to inject spatial and temporal awareness inside BERT core for a better and more accurate support for trajectory analysis tasks. One of the components that could be a key to injecting spatial and temporal awareness into BERT is the *loss function*. This function is responsible for evaluating BERT predictions during training along with penalizing or rewarding the model accordingly. BERT is trained by two tasks: **Masked Language Model (MLM)**, which randomly covers words in a statement and attempts to guess them, and **Next Sentence Prediction (NSP)**, which picks two sentences randomly and attempts to guess whether the second sentence follows the first one. The training loss function that BERT uses is the sum of the mean MLM likelihood and the mean NSP likelihood. The first part of the loss function (MLM likelihood) is a kind of binary, where only predictions that exactly match the masked word are considered correct and rewarded, while all other predictions are considered incorrect and penalized. There is no notion that some words are closer to the correct answer than others. The model adjusts its parameters toward the correct one and hopefully comes back in the next iteration and guesses it correctly. Applying this training task in the case of trajectories using GPS points instead of words means that predicting a GPS point that does not exactly match the masked/covered point is treated as a completely wrong prediction. This also means that predicting a point that is a few feet away from the actual point is as wrong as predicting a point that is miles away, because both penalize the model the same way. This is mainly because the loss function (and BERT) has no spatial awareness, and hence there is no sense of being almost close to the correct answer. As a result, this makes it hard for the model to learn unless it guesses correctly, which is almost impossible given the trajectory characteristics such as the presence of noisy data and the small ratio of training examples compared to the number of possible GPS points. To overcome this issue, we can adjust the MLM likelihood part of the loss function via an additional spatial penalty component that correlates with this distance. In particular, it ensures that the further the prediction from the actual point the higher the penalty. By implementing this approach, the model is encouraged to closely match the actual points and receives cues about its proximity to the correct answer. Such spatial guidance helps the model to overcome the noisiness and lack of large training examples as it would help in converging with good accuracy even with small training data.

## 6 INITIAL RESULTS

This section provides initial results that show the promise of our vision. In particular, we are doing exploratory evaluation for only the first direction of our vision, i.e., using the same architecture of BERT but customizing the trajectories and the way we call BERT (Section 5). We use our initial implementation, termed TrajBERT, to perform the evaluation on two trajectory analysis tasks, namely *trajectory imputation* (Section 3.1) and *trajectory prediction* (Section 3.2). For the training and evaluation dataset, we use a real trajectory dataset from the city of Porto, Portugal [99], composed of 1.7M trajectories for real taxi trips covering ~83M GPS points, driven for a total length of ~8.8M km spanning an area of ~500 km$^2$. We use 80% of these trajectories for training and keep the remaining 20% for testing. In the case of trajectory imputation task, we sparsify the testing trajectories by keeping the first point in the trajectory, then removing all points within distance $Sparse_{gap}$, then keeping the next point, and so on. Hence, all testing trajectories have gap distance $Sparse_{gap}$ between each two consecutive GPS points. We compare TrajBERT against a baseline approach, termed linear interpolation, that will just impute each trajectory segment by a straight line between its two end points. In the case of trajectory prediction, we convert each testing trajectory into two sub-trajectories by splitting it in the middle. Then, we use the first half as the input for the prediction task to predict the next sub-trajectory of length $Prediction_{Length}$. We use the second half, which is the ground truth in this case, to evaluate the predicted sub-trajectory. We compare TrajBERT against a baseline approach, termed linear extrapolation, that will just extend a straight line at the end of the input trajectory using the direction of its last two points.

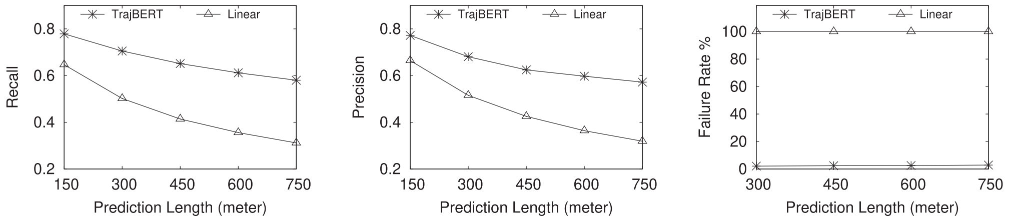
We use three performance measures, *recall*, *precision*, and *failure rate*. To measure the *recall* and *precision*, we discretize the ground-truth trajectory into $G$ artificial points by placing one point each 100 m. Similarly, we discretize the imputed/predicted trajectory into $I$ artificial points by placing one point each 100 m. Hence, the *recall* is computed as the ratio of points in $G$ that are correctly recovered/recalled within an accuracy threshold $\delta = 25$ m from the imputed/predicted trajectory. The higher the recall the more accurate is the algorithm. The *precision* is computed as the ratio of points in $I$ that are within an accuracy threshold $\delta = 25$ m from the ground truth. The higher the precision the more accurate the algorithm. For the *failure rate*, an algorithm fails when it does not return any point for imputation or prediction. For TrajBERT, this can happen when it employs spatial constraints on BERT output as a form of customizing the use of BERT as described in direction I in our vision (Section 5). For example, a spatial constraint may reject any imputation point that is further than a certain distance from the trajectory (see Reference [89] for details on spatial constraints). When no BERT output passes the constraints, we resort to a linear interpolation/extrapolation solution and count this as one failure for TrajBERT. Hence, the failure rate is the ratio of trajectory segments that we could not properly impute or the ratio of trajectories that we could not properly predict their next sub-trajectory. With this, the baseline linear interpolation/extrapolation methods have a 100% failure rate, as they always resort to naive linear lines.

Figure 6 gives our initial experiments for trajectory imputation task, which show the impact of varying $Sparse_{gap}$ from 500 to 4,000 m on both TrajBERT and linear interpolation with respect to the three performance measures. Figure 6(a) gives the *recall* ratio of both TrajBERT and linear interpolation. TrajBERT consistently outperforms the linear interpolation baseline with 2× to 7× better recall. For example, at a sparsity gap of 1,000 m, the TrajBERT recall is 0.7, which is three times better than the recall for linear interpolation. The performance increase is getting higher



(a) Recall (b) Precision (c) Failure Rate

Fig. 6. Performance results for imputation task.



(a) Recall (b) Precision (c) Failure Rate

Fig. 7. Performance results for prediction task.

with large sparse gaps, where for a 4,000-m gap, TrajBERT recall is 0.43, while linear interpolation recall is only 0.06. This shows that TrajBERT is more resilient to larger gaps. Figure 6(b) gives the *precision* ratio of both TrajBERT and linear interpolation. The results for the *precision* exhibits similar performance to that of the *recall*, where TrajBERT consistently outperforms linear interpolation in both short and large sparsity gaps. Figure 6(c) gives the failure rate of TrajBERT compared to the 100% linear interpolation basesline. TrajBERT failure rate is almost 0% regardless of the sparseness, meaning that it is always capable of finding imputation points between the two segment end points. This is mainly because our training dataset is kind of more dense than typical trajectory data. TrajBERT failure rate would definitely go up if the dataset is more sparse. However, the experiment shows that if given a good trajectory dataset, then TrajBERT can have almost 0% failure rate.

Figure 7 depicts our initial experiments for trajectory prediction task, which show the impact of varying $Prediction_{Length}$ from 150 to 750 m on both TrajBERT and linear extrapolation. Figure 7(a) gives the *recall* ratio and shows that TrajBERT consistently outperforms linear extrapolation at all values of $Prediction_{Length}$. Although the next sub-trajectory at small prediction lengths such as 150 and 300 m is highly likely to remain as a straight line or at least for a major portion of it, TrajBERT still outperforms the linear extrapolation even in such cases. For example, at $Prediction_{Length}$ = 300 m, TrajBERT recall is 0.7 compared to only 0.5 for linear extrapolation, which is a 40% performance gain. This performance gain of TrajBERT increases as we increase the length of the predicted next sub-trajectory. For example, at $Prediction_{Length}$ = 450 m, TrajBERT achieves a 60% performance gain (recall is 0.65 for TrajBERT and 0.4 for linear extrapolation). This is because the next sub-trajectory in this case is likely to have more curves and turns, which TrajBERT was able to predict more accurately. It becomes even more evident with larger prediction lengths such as 600 and 750 m as TrajBERT achieves around 90% of performance gain (at $Prediction_{Length}$ = 750 m, recall is 0.59 for TrajBERT and 0.31 for linear extrapolation). This shows the resilience of TrajBERT and its ability to predict relatively large sub-trajectories with more turns. Figure 7(b) gives the precision ratio, which exhibits a very similar performance as the recall. Figure 7(c) gives the failure rate, which is around 2% for TrajBERT regardless of the prediction length. This is likely due to the lack of enough training data in certain areas. However, the experiment shows that given a good real trajectory dataset like the one used here, TrajBERT would be able to predict points for the next sub-trajectory in almost all cases.

A detailed architecture of TrajBERT is explained in our recently published papers [88, 89], which include extensive experiments for the imputation task, as well as evaluation against more state-of-the-art baselines. Though the experiments here are initial and not conclusive, as they are made for only two trajectory operations, one dataset, and not trying to vary many parameters, they still show the promise of our vision. The objective here is to show that there is a road to our vision, and even a strawman implementation of some initial ideas would achieve highly promising results.

## 7 RELATED WORK

This section discusses related work to our vision. BERT [29] language model utilizes an architecture known as the Transofrmer [120], which encodes the input text into numeric vectors and uses such encodings to solve NLP tasks. Many BERT variants and other language models have later emerged (e.g., DistilBERT [106], RoBERTa [70], ALBERT [56], and ELECTRA [20]) adopting similar design of BERT and following its idea in solving many NLP tasks. Other language models (e.g., GPT [11], T5 [102], and BART [57]) have employed the transformer architecture in the decoding process that emits the text, which made such models excel more in tasks of generative nature such as text summarization and translation. Without loss of generality, we use BERT in this article as an example of a language model capable of performing various NLP tasks, which had a major impact and shift in the NLP community. However, given that the similarities between trajectories and statements (Section 2) still hold regardless of the NLP model used, our ideas presented in this article are not limited to BRET and would generally still apply to other NLP models. Since we have already discussed related work to each trajectory operation in Section 3, we are limiting the discussion in this section to those studies that aim to deploy ideas from the NLP domain to spatial data. We are classifying such work into the two below categories:

**Pre-BERT Era: Embedding and Representation Learning.** Many researchers in the spatial domain were inspired by the discovery of Word Embedding in NLP, with its classical and popular Word2Vec approach, proposed in 2013 [83]. The main idea of Word2Vec is to map each vocabulary word to a numerical vector that represents it and thus enable computations and mathematical operations on words. Hence, several efforts in the spatial community have taken similar approaches for various spatial datasets (e.g., see References [35, 68, 78, 123, 139, 140, 147, 155, 156]). Many of these approaches have even named their techniques in an analogous way to Word2Vec, e.g., GPS2Vec [141], Loc2Vec [107, 129], Location2Vec [157], Place2Vec [135], and POI2Vec [34]. The main goal of these studies is to learn and obtain embedding (i.e., vectors or numerical representations) for a wide range of spatial elements such as GPS points [141], points of interest PoI(s) [78, 107, 129, 135, 156], trajectories [35, 68, 155], road network nodes and edges [18], andgeographical areas and regions such as neighborhoods, cities, or zones [123, 140, 147, 157]. The term embedding refers to the fact that some of the hidden information about the element (e.g., meaning and semantics in the case of a vocabulary word, or location type in the case of a PoI or GPS point) are discovered and learned via deep neural networks and then embedded/encoded in a numerical form in the vector representation. Despite the importance of the embedding process, it is limited to basic element-wise operations such as computing the difference or similarity between two words or locations. Embedding alone is not sufficient for more complex tasks such as the next sentence prediction or next trajectory prediction. This is why all the efforts for spatial data embedding are mainly concerned with a very specific task and cannot be extended to other tasks. For example, Place2Vec is only concerned with place type similarity [135], SERM and Loc2Vec [107, 139] are only concerned with next point prediction, while another work [68] is only concerned with activity similarity.

**The BERT Era: Language Models for Spatial Data.** Due to the limitations of the word embedding approach, the NLP community came up with the idea of BERT as a language model that is able to process complete documents and sentences. Then, learning the embedding becomes only one component among many other components of BERT. Research efforts in employing language models for the spatial domain are mainly geared toward utilizing it, as a *pretrained* model, in its lingual form by verbally asking it questions of spatial nature [44, 97, 134]. The main idea is to use language as a medium to make the model perform certain tasks such as correcting an address or forecasting the name of the next PoI from an input sentence. This is the opposite of utilizing the model architecture only and then training it from scratch on spatial data. Hence, this approach is limited by the ability to verbally represent the data. It is also subject to the model ability to understand and comprehend geospatial semantics from the text. To overcome these limitations, a follow-up work is proposed to add external supervised training [44, 97], for example, using existing spatial datasets and spatial knowledge graphs to artificially generate new sentences about places, PoIs, distances between them, their locations, types, and add these new sentences to the training corpus in an effort to increase the language model ability to do spatial reasoning from language. A recent work that falls in this category is SpaBERT [64], which trains BERT with pseudo sentences generated from large geographic datasets. SpaBERT is dedicated to generating a better representation for geo-entities mentioned in the text, such that the representation depends on nearby geo-entities that may exist within the sentence, hence improving BERT ability to understand spatial entities and spatial semantics in text. A more recent work has investigated using language models for spatial data of multiple forms [77]. In particular, it looks at how to combine multiple forms of datasets (e.g., satellite images, graphs, text) that contain spatial data and then use this combination to train a geospatial model for various spatial tasks. With respect to trajectory analysis, few research efforts have recently considered using a language model like BERT as is for various operations, including similarity [28, 61], prediction [12, 37, 66], imputation [21, 101], and classification [65].

**Our Vision: BERT-like Models for Trajectory Analysis** Our vision goes beyond word embedding and focuses more on BERT and NLP models. Unlike all existing work, our vision is not to improve BERT spatial understanding. Instead, our vision is to use the architecture of BERT and train it on trajectories rather than language. Our vision is not concerned with only one specific trajectory operation. Instead, our vision aims to build a BERT-like model that can support a wide variety of trajectory analysis tasks. This goes along with the spirit of BERT itself that was not designed for one specific NLP task. Instead, it is made to support a wide variety of NLP tasks.

## 8 CONCLUSION

We have presented our vision to utilize state-of-the-art language models in NLP, e.g., BERT, to support trajectory analysis operations. The promise is that doing so will pave the way toward a long-waited full-fledged trajectory data management system that natively supports all trajectory analysis operations. We show that trajectories in a space exhibit a very similar behavior to statements in a language, and hence many of the NLP tasks on statements are analogous to spatial analysis tasks on trajectories. However, we also point out several challenges that hinder the applicability of BERT as it is to trajectory analysis. This helps in outlining the road to realizing our vision by exploiting two orthogonal, but complementary, directions. The first direction is to customize both trajectory data and BERT usage to match each other, without the need to change the core of BERT. The second direction is to inject spatial and temporal awareness inside the core of BERT itself to achieve better results. Initial exploratory results for one trajectory operation and one dataset confirm the promise and possibility of our vision.

## REFERENCES

[1] ACM SIGSPATIAL Cup 2017. Retrieved from http://sigspatial2017.sigspatial.org/giscup2017/download

[2] Louai Alarabi, Mohamed F. Mokbel, and Mashaal Musleh. 2018. ST-Hadoop: A MapReduce framework for spatio-temporal data. *GeoInformatica* 22, 4 (2018), 785–813.

[3] Rakan Alseghayer. 2021. Racoon: Rapid contact tracing of moving objects using smart indexes. In *MDM*. 274–276.

[4] Akinori Asahara, Kishiko Maruyama, Akiko Sato, and Kouichi Seto. 2011. Pedestrian-movement prediction based on mixed markov-chain model. In *SIGSPATIAL*. 25–33.

[5] Emmanouil Barmpounakis and Nikolas Geroliminis. 2020. On the new era of urban traffic monitoring with massive drone data: The pNEUMA large-scale field experiment. *Transp. Res. Part C: Emerg. Technol.* 111 (2020), 50–71.

[6] Wendy Bohte and Kees Maat. 2009. Deriving and validating trip purposes and travel modes for multi-day gps-based travel surveys: A large-scale application in the netherlands. *Transp. Res. Part C: Emerg. Technol.* 17, 3 (2009), 285–297.

[7] Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi. 2014. CRAWDAD dataset roma/taxi (v. 2014-07-17). Retrieved from https://crawdad.org/roma/taxi/20140717

[8] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. 2005. On map-matching vehicle tracking data. In *VLDB*. 853–864.

[9] Igo Ramalho Brilhante, José Antônio Fernandes de Macêdo, Franco Maria Nardini, Raffaele Perego, and Chiara Renso. 2015. Planning sightseeing tours using crowdsensed trajectories. *ACM SIGSPATIAL Spec.* 7, 1 (2015), 59–66.

[10] Isaac Brodsky. H3: Uber's Hexagonal Hierarchical Spatial Index. Retrieved from https://eng.uber.com/h3/

[11] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

[12] Dun Cao, Kai Zeng, Jin Wang, Pradip Kumar Sharma, Xiaomin Ma, Yonghe Liu, and Siyuan Zhou. 2022. BERT-based deep spatial-temporal network for taxi demand prediction. *IEEE Trans. Intell. Transp. Syst.* 23, 7 (2022), 9442–9454.

[13] Mário Cardoso, André Cavalheiro, Alexandre Borges, Ana Filipa Duarte, Amílcar Soares, Maria João Pereira, Nuno Jardim Nunes, Leonardo Azevedo, and Arlindo L. Oliveira. 2022. Modeling the geospatial evolution of COVID-19 using spatio-temporal convolutional sequence-to-sequence neural networks. *ACM Trans. Spatial Algor. Syst.* 8, 4 (2022), 28:1–28:19.

[14] Chao Chen, Shuhai Jiao, Shu Zhang, Weichen Liu, Liang Feng, and Yasha Wang. 2018. TripImputor: Real-time imputing taxi trip purpose leveraging multi-sourced urban data. *IEEE Trans. Intell. Transport. Syst.* 19, 10 (2018), 3292–3304.

[15] Lisi Chen, Shuo Shang, Christian S. Jensen, Bin Yao, and Panos Kalnis. 2020. Parallel semantic trajectory similarity join. In *ICDE*. 997–1008.

[16] Lisi Chen, Shuo Shang, Christian S. Jensen, Bin Yao, Zhiwei Zhang, and Ling Shao. 2019. Effective and efficient reuse of past travel behavior for route recommendation. In *KDD*. 488–498.

[17] Yen-Chun Chen, Zhe Gan, Yu Cheng, Jingzhou Liu, and Jingjing Liu. 2020. Distilling knowledge learned in BERT for text generation. In *ACL*. 7893–7905.

[18] Yile Chen, Xiucheng Li, Gao Cong, Zhifeng Bao, Cheng Long, Yiding Liu, Arun Kumar Chandran, and Richard Ellison. 2021. Robust road network representation learning: When traffic patterns meet traveling semantics. In *CIKM*. 211–220.

[19] Yuhao Chen and Farhana H. Zulkernine. 2021. BIRD-QA: A BERT-based information retrieval approach to domain specific question answering. In *IEEE Big Data*. 3503–3510.

[20] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

[21] Alessandro Crivellari, Bernd Resch, and Yuhui Shi. 2022. TraceBERT - A feasibility study on reconstructing spatial-temporal gaps from incomplete motion trajectories via BERT training process on discrete location sequences. *Sensors* 22, 4 (2022), 1682.

[22] Danilo Croce, Giuseppe Castellucci, and Roberto Basili. 2020. GAN-BERT: Generative adversarial learning for robust text classification with a bunch of labeled examples. In *ACL*. 2114–2119.

[23] Philippe Cudré-Mauroux and Eugene Wu andvSamuel Madden. 2010. TrajStore: An adaptive storage system for very large trajectory data sets. In *ICDE*. 109–120.

[24] Sajal K. Das, Diane J. Cook, Amiya Bhattacharya, Edwin O. Heierman III, and Tze-Yun Lin. 2002. The role of prediction algorithms in the mavhome smart home architecture. *IEEE Wireless Commun.* 9, 6 (2002), 77–84.

[25] Sajal K. Das and Sanjoy K. Sen. 1999. Adaptive location prediction strategies based on a hierarchical network model in a cellular mobile environment. *Comput. J.* 42, 6 (1999), 473–486.

[26] Elton Figueiredo de S. Soares, Kate Revoredo, Fernanda Baião, Carlos Alvaro de M. S. Quintella, and Carlos Alberto Vieira Campos. 2019. A combined solution for real-time travel mode detection and trip purpose prediction. *IEEE Trans. Intell. Transp. Syst.* 20, 12 (2019), 4655–4664.

[27] Roniel S. de Sousa, Azzedine Boukerche, and Antonio A. F. Loureiro. 2020. Vehicle trajectory similarity: Models, methods, and applications. *ACM Comput. Surv.* 53, 5 (2020), 94:1–94:32.

[28] Liwei Deng, Hao Sun, Rui Sun, Yan Zhao, and Han Su. 2022. Efficient and effective similar subtrajectory search: A spatial-aware comprehension approach. *ACM Trans. Intell. Syst. Technol.* 13, 3 (2022), 35:1–35:22.

[29] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

[30] Daniel Dias and Luis Henrique Maciel Kosmalski Costa. 2018. CRAWDAD dataset coppe-ufrj/RioBuses (v. 2018-03-19). Retrieved from https://crawdad.org/coppe-ufrj/RioBuses/20180319

[31] Chunning Du, Haifeng Sun, Jingyu Wang, Qi Qi, and Jianxin Liao. 2020. Adversarial and domain-aware BERT for cross-domain sentiment analysis. In *ACL*. 4019–4028.

[32] Mohamed M. Elshrif, Keivin Isufaj, and Mohamed F. Mokbel. 2022. Network-less trajectory imputation. In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM SIGSPATIAL GIS*.

[33] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. 2018. DeepMove: Predicting human mobility with attentional recurrent networks. In *WWW*. 1459–1468.

[34] Shanshan Feng, Gao Cong, Bo An, and Yeow Meng Chee. 2017. POI2Vec: Geographical latent representation for predicting future visitors. In *AAAI*. 102–108.

[35] Tao-Yang Fu and Wang-Chien Lee. 2020. Trembr: Exploring road networks for trajectory representation learning. *ACM Trans. Intell. Syst. Technol.* 11, 1 (2020), 10:1–10:25.

[36] Qiang Gao, Goce Trajcevski, Fan Zhou, Kunpeng Zhang, Ting Zhong, and Fengli Zhang. 2019. DeepTrip: Adversarially understanding human mobility for trip recommendation. In *SIGSPATIAL*. 444–447.

[37] Francesco Giuliari, Irtiza Hasan, Marco Cristani, and Fabio Galasso. 2020. Transformer networks for trajectory forecasting. In *ICPR*. 10335–10342.

[38] Marta C. González, César A. Hidalgo, and Albert-László Barabási. 2008. Understanding individual human mobility patterns. *Nature* 453, 7196 (2008), 779–782.

[39] Songtao He, Favyen Bastani, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, and Sam Madden. 2018. Roadrunner: Improving the precision of road network inference from GPS trajectories. In *SIGSPATIAL*. 3–12.

[40] Tianfu He, Jie Bao, Sijie Ruan, Ruiyuan Li, Yanhua Li, Hui He, and Yu Zheng. 2020. Interactive bike lane planning using sharing bikes' trajectories. *IEEE Trans. Knowl. Data Eng.* 32, 8 (2020), 1529–1542.

[41] Abdeltawab Hendawi and Mohamed F. Mokbel. 2012. Panda: A predictive spatio-temporal query processor. In *SIGSPATIAL*. 13–22.

[42] Abdeltawab Hendawi and Mohamed F. Mokbel. 2012. Predictive spatio-temporal queries: A comprehensive survey and future directions. In *MobiGIS/SIGSPATIAL GIS*. 97–104.

[43] Bushra Hossain, Kazi Abir Adnan, Md. Fazle Rabbi, and Mohammed Eunus Ali. 2020. Modelling road traffic congestion from trajectories. In *DSIT*. 117–122.

[44] Jizhou Huang, Haifeng Wang, Yibo Sun, Yunsheng Shi, Zhengjie Huang, An Zhuo, and Shikun Feng. 2022. ERNIE-GeoL: A geography-and-language pre-trained model and its applications in baidu maps. In *KDD*. 3029–3039.

[45] Xiaocheng Huang, Yifang Yin, Simon Lim, Guanfeng Wang, Bo Hu, Jagannadan Varadarajan, Shaolin Zheng, Ajay Bulusu, and Roger Zimmermann. 2019. Grab-posisi: An extensive real-life GPS trajectory dataset in southeast asia. In *PredictGIS*. 1–10.

[46] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, and Christian S. Jensen. 2010. Path prediction and predictive range querying in road network databases. *VLDB J.* 19, 4 (2010), 585–602.

[47] Antonios Karatzoglou, Adrian Jablonski, and Michael Beigl. 2018. A Seq2Seq learning approach for modeling semantic trajectories and predicting the next location. In *SIGSPATIAL*. 528–531.

[48] Panagiota Katsikouli, Rik Sarkar, and Jie Gao. 2014. Persistence based online signal and trajectory simplification for mobile devices. In *SIGSPATIAL*. 371–380.

[49] Amin Vahedian Khezerlou, Xun Zhou, Ling Tong, Yanhua Li, and Jun Luo. 2021. Forecasting gathering events through trajectory destination prediction: A dynamic hybrid model. *IEEE Trans. Knowl. Data Eng.* 33, 3 (2021), 991–1004.

[50] Sang-Wook Kim, Jung-Im Won, Jong-Dae Kim, Miyoung Shin, Junghoon Lee, and Hanil Kim. 2007. Path prediction of moving objects on road networks through analyzing past trajectories. In *KES*, Vol. 4692. 379–389.

[51] Satoshi Koide, Yukihiro Tadokoro, Chuan Xiao, and Yoshiharu Ishikawa. 2018. CiNCT: Compression and retrieval for massive vehicular trajectories via relative movement labeling. In *ICDE*. 1097–1108.

[52] Benjamin B. Krogh, Ove Andersen, Edwin Lewis-Kelham, Nikos Pelekis, Yannis Theodoridis, and Kristian Torp. 2013. Trajectory based traffic analysis. In *SIGSPATIAL*. 526–529.

[53] Chia-Chih Kuo, Shang-Bao Luo, and Kuan-Yu Chen. 2020. An audio-enriched BERT-based framework for spoken multiple-choice question answering. In *Interspeech*. 4173–4177.

[54] Moritz Laass, Marie Kiermeier, and Martin Werner. 2021. Improving persistence based trajectory simplification. In *MDM*. 157–162.

[55] Hai Lan, Jiong Xie, Zhifeng Bao, Feifei Li, Wei Tian, Fang Wang, Sheng Wang, and Ailin Zhang. 2022. VRE: A versatile, robust, and economical trajectory data system. *Proc. VLDB* 15, 12 (2022), 3398–3410.

[56] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *ICLR*.

[57] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*. 7871–7880.

[58] Bozhao Li, Zhongliang Cai, Mengjun Kang, Shiliang Su, Shanshan Zhang, Lili Jiang, and Yong Ge. 2021. A trajectory restoration algorithm for low-sampling-rate floating car data and complex urban road networks. *Int. J. GIS* 35, 4 (2021), 717–740.

[59] Bing Li, Yukai Miao, Yaoshu Wang, Yifang Sun, and Wei Wang. 2021. Improving the efficiency and effectiveness for BERT-based entity resolution. In *AAAI*. 13226–13233.

[60] Tianyi Li, Ruikai Huang, Lu Chen, Christian S. Jensen, and Torben Bach Pedersen. 2020. Compression of uncertain trajectories in road networks. *Proc. VLDB* 13, 7 (2020), 1050–1063.

[61] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S. Jensen, and Wei Wei. 2018. Deep representation learning for trajectory similarity computation. In *ICDE*. 617–628.

[62] Yang Li, Yangyan Li, Dimitrios Gunopulos, and Leonidas J. Guibas. 2016. Knowledge-based trajectory completion from sparse GPS samples. In *SIGSPATIAL*. 33:1–33:10.

[63] Yanhua Li, Jun Luo, Chi-Yin Chow, Kam-Lam Chan, Ye Ding, and Fan Zhang. 2015. Growing the charging station network for electric vehicles with trajectory data analytics. In *ICDE*. 1376–1387.

[64] Zekun Li, Jina Kim, Yao-Yi Chiang, and Muhao Chen. 2022. SpaBERT: A pretrained language model from geographic data for geo-entity representation. In *EMNLP*. 2757–2769.

[65] Yuxuan Liang, Kun Ouyang, Yiwei Wang, Xu Liu, Hongyang Chen, Junbo Zhang, Yu Zheng, and Roger Zimmermann. 2022. TrajFormer: Efficient trajectory classification with transformers. In *CIKM*. 1229–1237.

[66] Yan Lin, Huaiyu Wan, Shengnan Guo, and Youfang Lin. 2021. Pre-training context and time aware location embeddings from spatial-temporal trajectories for user next location prediction. In *AAAI*. 4241–4248.

[67] Jonathan Liono, Zahraa S. Abdallah, A. Kai Qin, and Flora D. Salim. 2018. Inferring transportation mode and human activity from mobile sensing in daily life. In *MobiQuitous*. 342–351.

[68] An Liu, Yifan Zhang, Xiangliang Zhang, Guanfeng Liu, Yanan Zhang, Zhixu Li, Lei Zhao, Qing Li, and Xiaofang Zhou. 2022. Representation learning with multi-level attention for activity trajectory similarity computation. *IEEE Trans. Knowl. Data Eng.* 34, 5 (2022), 2387–2400.

[69] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the next location: A recurrent model with spatial and temporal contexts. In *AAAI*. 194–200.

[70] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[71] Cheng Long, Raymond Chi-Wing Wong, and H. V. Jagadish. 2013. Direction-preserving trajectory simplification. *Proc. VLDB* 6, 10 (2013), 949–960.

[72] Cheng Long, Raymond Chi-Wing Wong, and H. V. Jagadish. 2014. Trajectory simplification: On minimizing the direction-based error. *Proc. VLDB* 8, 1 (2014), 49–60.

[73] Jed A. Long. 2016. Kinematic interpolation of movement data. *Int. J. Geogr. Inf. Sci.* 30, 5 (2016), 854–868.

[74] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. 2009. Map-matching for low-sampling-rate GPS trajectories. In *SIGSPATIAL*. 352–361.

[75] Suxing Lyu, Tianyang Han, Yuuki Nishiyama, Kaoru Sezaki, and Takahiko Kusakabe. 2022. A plug-in memory network for trip purpose classification. In *SIGSPATIAL*. 34:1–34:12.

[76] Suxing Lyu and Takahiko Kusakabe. 2021. Graph-aware chained trip purpose inference. In *ITSC*. 3691–3697.

[77] Gengchen Mai, Chris Cundy, Kristy Choi, Yingjie Hu, Ni Lao, and Stefano Ermon. 2022. Towards a foundation model for geospatial artificial intelligence (vision paper). In *SIGSPATIAL*. 106:1–106:4.
[78] Gengchen Mai, Krzysztof Janowicz, Bo Yan, Rui Zhu, Ling Cai, and Ni Lao. 2020. Multi-scale representation learning for spatial feature distributions using grid cells. In *ICLR*.
[79] Mapillary. Unveiling the Mapping in Logistics Report: The Impact of Broken Maps on Last-Mile Deliveries. Retrieved from https://blog.mapillary.com/update/2020/02/14/mapping-in-logistics.html
[80] Wesley Mathew, Ruben Raposo, and Bruno Martins. 2012. Predicting future locations with hidden Markov models. In *Ubicomp*. 911–918.
[81] Chuishi Meng, Xiuwen Yi, Lu Su, Jing Gao, and Yu Zheng. 2017. City-wide traffic volume inference with loop detector data and taxi trajectories. In *SIGSPATIAL*. 1:1–1:10.
[82] MicrosoftMissingRoads. Discover New Roads with Bing Maps. Retrieved from https://blogs.bing.com/maps/2022-12/Bing-Maps-is-bringing-new-roads/
[83] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR*.
[84] Mohamed F. Mokbel, Sofiane Abbar, and Rade Stanojevic. 2020. Contact tracing: Beyond the apps. *ACM SIGSPATIAL Spec.* 12, 2 (2020), 15–24.
[85] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. 2009. Wherenext: A location predictor on trajectory pattern mining. In *SIGKDD*. 637–646.
[86] David Montoya, Serge Abiteboul, and Pierre Senellart. 2015. Hup-Me: Inferring and reconciling a timeline of user activity from rich smartphone data. In *SIGSPATIAL*. 62:1–62:4.
[87] Mashaal Musleh, Sofiane Abbar, Rade Stanojevic, and Mohamed F. Mokbel. 2021. QARTA: An ML-based system for accurate map services. *Proc. VLDB* 14, 11 (2021), 2273–2282.
[88] Mashaal Musleh and Mohamed Mokbel. 2023. A demonstration of KAMEL: A scalable BERT-based system for trajectory imputation. In *SIGMOD*.
[89] Mashaal Musleh and Mohamed Mokbel. 2023. KAMEL: A scalable BERT-based system for trajectory imputation. *Proc. VLDB* 17, 3 (2023), 525–538.
[90] Mashaal Musleh and Mohamed F. Mokbel. 2022. A demonstration of RASED: A scalable dashboard for monitoring road network updates in OSM. In *ICDE*. 3164–3149.
[91] Mashaal Musleh and Mohamed F. Mokbel. 2022. RASED: A scalable dashboard for monitoring road network updates in OSM. In *MDM*. 214–221.
[92] Mashaal Musleh, Mohamed F. Mokbel, and Sofiane Abbar. 2022. Let's speak trajectories. In *SIGSPATIAL*. 37:1–37:4.
[93] Anthony J. Nicholson and Brian D. Noble. 2008. BreadCrumbs: Forecasting mobile connectivity. In *MOBICOM*. 46–57.
[94] NYC. Kaggle. New York City Taxi Trip Duration. Retrieved from https://www.kaggle.com/c/nyc-taxi-trip-duration/data
[95] Simon Aagaard Pedersen, Bin Yang, and Christian S. Jensen. 2020. Anytime stochastic routing with hybrid learning. *Proc. VLDB* 13, 9 (2020), 1555–1567.
[96] Nicole Peinelt, Dong Nguyen, and Maria Liakata. 2020. tBERT: Topic models and BERT joining forces for semantic similarity detection. In *ACL*. 7047–7055.
[97] Vamsi Krishna Penumadu, Nitesh Methani, and Saurabh Sohoney. 2022. Learning geospatially aware place embeddings via weak-supervision. In *SIGSPATIAL*. 80:1–80:10.
[98] Lucas May Petry, Carlos Andres Ferrero, Luis Otávio Alvares, Chiara Renso, and Vania Bogorny. 2019. Towards semantic-aware multiple-aspect trajectory similarity measuring. *Trans. GIS* 23, 5 (2019), 960–975.
[99] Porto. Taxi Service Trajectory. Prediction Challenge. ECML PKDD 2015. Retrieved from http://www.geolink.pt/ecmlpkdd2015-challenge/dataset.html
[100] Reinald Adrian Pugoy and Hung-Yu Kao. 2021. Unsupervised extractive summarization-based representations for accurate and explainable collaborative filtering. In *ACL*. 2981–2990.
[101] Kyle Kai Qin, Yongli Ren, Wei Shao, Brennan Lake, Filippo Privitera, and Flora D. Salim. 2023. Multiple-level point embedding for solving human trajectory imputation with prediction. *ACM Trans. Spatial Algor. Syst.* (Feb. 2023).
[102] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67.
[103] Jinmeng Rao, Song Gao, and Xiaojin Zhu. 2021. VTSV: A privacy-preserving vehicle trajectory simulation and visualization platform using deep reinforcement learning. In *GeoAI*.
[104] Sasank Reddy, Min Y. Mun, Jeff Burke, Deborah Estrin, Mark H. Hansen, and Mani B. Srivastava. 2010. Using mobile phones to determine transportation modes. *ACM Trans. Sens. Netw.* 6, 2 (2010), 13:1–13:27.
[105] Amin Sadri, Flora D. Salim, Yongli Ren, Wei Shao, John Krumm, and Cecilia Mascolo. 2018. What will you do for the rest of the day?: An approach to continuous trajectory prediction. *Proc. ACM Interact. Mob. Wear. Ubiq. Technol.* 2, 4 (2018), 186:1–186:26.

[106] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. arXiv:1910.01108. Retrieved from http://arxiv.org/abs/1910.01108

[107] Abdessamed Sassi, Mohammed Brahimi, Walid Bechkit, and Abdelmalik Bachir. 2019. Location embedding and deep convolutional neural networks for next location prediction. In *IEEE LCN*. 149–157.

[108] Zeyuan Shang, Guoliang Li, and Zhifeng Bao. 2018. DITA: Distributed in-memory trajectory analytics. In *SIGMOD*. ACM, 725–740.

[109] Li Shen and Peter R Stopher. 2013. A process for trip purpose imputation from global positioning system data. *Transport. Res. Part C: Emerg. Technol.* 36 (2013), 261–267.

[110] Libo Song, David Kotz, Ravi Jain, and Xiaoning He. 2006. Evaluating next-cell predictors with extensive Wi-Fi mobility data. *IEEE Trans. Mob. Comput.* 5, 12 (2006), 1633–1649.

[111] Rade Stanojevic, Sofiane Abbar, Saravanan Thirumuruganathan, Sanjay Chawla, Fethi Filali, and Ahid Aleimat. 2018. Robust road map inference through network alignment of trajectories. In *SDM*. 135–143.

[112] Leon Stenneth, Ouri Wolfson, Philip S. Yu, and Bo Xu. 2011. Transportation mode detection using mobile phones and GIS information. In *SIGSPATIAL*. 54–63.

[113] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. 2020. A survey of trajectory distance measures and performance evaluation. *VLDB J.* 29, 1 (2020), 3–32.

[114] Christoph Sydora, Faiza Nawaz, Leepakshi Bindra, and Eleni Stroulia. 2022. Building occupancy simulation and analysis under virus scenarios. *ACM Trans. Spatial Algor. Syst.* 8, 3 (2022), 1–20.

[115] Kohei Tanaka, Yasue Kishino, Tsutomu Terada, and Shojiro Nishio. 2009. A destination prediction method using driving contexts and trajectory for car navigation systems. In *SAC*. 190–195.

[116] Traffic Technology Today. Poor Maps Costing Delivery Companies US $6bn Annually. Retrieved from https://www.traffictechnologytoday.com/news/mapping/poor-maps-costing-delivery-companies-us6bn-annually.html

[117] Kazuki Tsunematsu, Johanes Effendi, Sakriani Sakti, and Satoshi Nakamura. 2020. Neural speech completion. In *Interspeech*. 2742–2746.

[118] UCR. UCR STAR: The UCR Spatio-temporal Active Repository. Retrieved from https://star.cs.ucr.edu

[119] Amin Vahedian, Xun Zhou, Ling Tong, Yanhua Li, and Jun Luo. 2017. Forecasting gathering events through continuous destination prediction on big trajectory data. In *SIGSPATIAL*. 34:1–34:10.

[120] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.

[121] Bijun Wang, Yulong Wang, Kun Qin, and Qizhi Xia. 2018. Detecting transportation modes based on LightGBM classifier from GPS trajectory data. In *Geoinformatics*. 1–7.

[122] Guang Wang, Xiuyuan Chen, Fan Zhang, Yang Wang, and Desheng Zhang. 2019. Experience: Understanding long-term evolving patterns of shared electric vehicle networks. In *MobiCom*. 1–12.

[123] Hongjian Wang and Zhenhui Li. 2017. Region representation learning via mobility flow. In *CIKM*. 237–246.

[124] Jingyuan Wang, Jiawei Jiang, Wenjun Jiang, Chao Li, and Wayne Xin Zhao. 2021. LibCity: An open library for traffic prediction. In *SIGSPATIAL*. 145–148.

[125] Jingyuan Wang, Ning Wu, Xinxi Lu, Wayne Xin Zhao, and Kai Feng. 2021. Deep trajectory recovery with fine-grained calibration using kalman filter. *IEEE Trans. Knowl. Data Eeng.* 33, 3 (2021), 921–934.

[126] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, and Gao Cong. 2021. A survey on trajectory data management, analytics, and learning. *ACM Comput. Surv.* 54, 2 (2021), 39:1–39:36.

[127] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *SIGKDD*. 25–34.

[128] Zheng Wang, Cheng Long, and Gao Cong. 2021. Trajectory simplification with reinforcement learning. In *ICDE*. 684–695.

[129] Yan Wen, Jiansong Zhang, Qingtian Zeng, Xin Chen, and Feng Zhang. 2019. Loc2Vec-based cluster-level transition behavior mining for successive POI recommendation. *IEEE Access* 7 (2019), 109311–109319.

[130] Hao Wu, Jiangyun Mao, Weiwei Sun, Baihua Zheng, Hanyuan Zhang, Ziyang Chen, and Wei Wang. 2016. Probabilistic robust route recovery with spatio-temporal dynamics. In *KDD*. 1915–1924.

[131] Guangnian Xiao, Zhicai Juan, and Chunqin Zhang. 2016. Detecting trip purposes from smartphone-based travel surveys with artificial neural networks and particle swarm imization. *Transport. Res. Part C: Emerg. Technol.* 71 (2016), 447–463.

[132] Li Xiong, Cyrus Shahabi, Yanan Da, Ritesh Ahuja, Vicki Hertzberg, Lance Waller, Xiaoqian Jiang, and Amy Franklin. 2020. REACT: Real-time contact tracing and risk monitoring using privacy-enhanced mobile tracking. *ACM SIGSPATIAL Spec.* 12, 2 (2020), 3–14.

[133] Hao Xue, Flora D. Salim, Yongli Ren, and Nuria Oliver. 2021. MobTCast: Leveraging auxiliary trajectory forecasting for human mobility prediction. In *NeurIPS*. 30380–30391.

[134] Hao Xue, Bhanu Prakash Voutharoja, and Flora D. Salim. 2022. Leveraging language foundation models for human mobility forecasting. In *SIGSPATIAL*. 90:1–90:9.

[135] Bo Yan, Krzysztof Janowicz, Gengchen Mai, and Song Gao. 2017. From ITDL to Place2Vec: Reasoning about place type similarity and relatedness by learning embeddings from augmented spatial contexts. In *SIGSPATIAL*. 35:1–35:10.

[136] Peilun Yang, Hanchen Wang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2021. T3S: Effective representation learning for trajectory similarity computation. In *ICDE*. 2183–2188.

[137] Xiaochun Yang, Bin Wang, Kai Yang, Chengfei Liu, and Baihua Zheng. 2018. A novel representation and compression for queries on trajectories in road networks. *IEEE Trans. Knowl. Data Eng*. 30, 4 (2018), 613–629.

[138] Yu Yang, Fan Zhang, and Desheng Zhang. 2018. SharedEdge: GPS-free fine-grained travel time estimation in state-level highway systems. *Proc. ACM Interact. Mobile Wear. Ubiq. Technol.* 2, 1 (2018), 48:1–48:26.

[139] Di Yao, Chao Zhang, Jian-Hui Huang, and Jingping Bi. 2017. SERM: A recurrent model for next location prediction in semantic trajectories. In *CIKM*. 2411–2414.

[140] Zijun Yao, Yanjie Fu, Bin Liu, Wangsu Hu, and Hui Xiong. 2018. Representing urban functions through zone embedding with human mobility patterns. In *IJCAI*. 3919–3925.

[141] Yifang Yin, Zhenguang Liu, Ying Zhang, Sheng Wang, Rajiv Ratn Shah, and Roger Zimmermann. 2019. GPS2Vec: Towards generating worldwide GPS embeddings. In *SIGSPATIAL*. 416–419.

[142] Josh Jia-Ching Ying, Wang-Chien Lee, Tz-Chiao Weng, and Vincent S. Tseng. 2011. Semantic trajectory mining for location prediction. In *SIGSPATIAL*. 34–43.

[143] Haitao Yuan and Guoliang Li. 2019. Distributed in-memory trajectory similarity search and join on road network. In *ICDE*. 1262–1273.

[144] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. 2010. T-drive: Driving directions based on taxi trajectories. In *SIGSPATIAL*. 99–108.

[145] Aoqian Zhang, Shaoxu Song, Jianmin Wang, and Philip S. Yu. 2017. Time series data cleaning: From anomaly detection to anomaly repairing. *Proc. VLDB* 10, 10 (2017), 1046–1057.

[146] Shaohua Zhang, Haoran Huang, Jicong Liu, and Hang Li. 2020. Spelling error correction with soft-masked BERT. In *ACL*. 882–890.

[147] Yunchao Zhang, Yanjie Fu, Pengyang Wang, Xiaolin Li, and Yu Zheng. 2019. Unifying inter-region autocorrelation and intra-region structures for spatial embedding via collective adversarial learning. In *SIGKDD*. 1700–1708.

[148] Yan Zhao, Shuo Shang, Yu Wang, Bolong Zheng, Quoc Viet Hung Nguyen, and Kai Zheng. 2018. REST: A reference-based framework for spatio-temporal trajectory compression. In *KDD*. 2797–2806.

[149] Guanjie Zheng, Hanyang Liu, Kai Xu, and Zhenhui Li. 2020. Learning to simulate vehicle trajectories from demonstrations. In *ICDE*.

[150] Kai Zheng, Yu Zheng, Xing Xie, and Xiaofang Zhou. 2012. Reducing uncertainty of low-sampling-rate trajectories. In *ICDE*. 1144–1155.

[151] Yu Zheng. 2015. Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol.* 6, 3 (2015), 29:1–29:41.

[152] Yu Zheng, Yukun Chen, Quannan Li, Xing Xie, and Wei-Ying Ma. 2010. Understanding transportation modes based on GPS data for web applications. *ACM Trans. Web* 4, 1 (2010), 1:1–1:36.

[153] Yu Zheng, Like Liu, Longhao Wang, and Xing Xie. 2008. Learning transportation mode from raw GPS data for geographic applications on the web. In *WWW*. 247–256.

[154] Fan Zhou, Hantao Wu, Goce Trajcevski, Ashfaq A. Khokhar, and Kunpeng Zhang. 2020. Semi-supervised trajectory understanding with POI attention for end-to-end trip recommendation. ACM Trans. Intell. Syst. Technol. 6, 2 (2020), 13:1–13:25.

[155] Ningnan Zhou, Wayne Xin Zhao, Xiao Zhang, Ji-Rong Wen, and Shan Wang. 2016. A general multi-context embedding model for mining human trajectory data. *IEEE Trans. Knowl. Data Eng*. 28, 8 (2016), 1945–1958.

[156] Yang Zhou and Yan Huang. 2018. DeepMove: Learning place representations through large scale movement data. In *IEEE Big Data*. 2403–2412.

[157] Minfeng Zhu, Wei Chen, Jiazhi Xia, Yuxin Ma, Yankong Zhang, Yuetong Luo, Zhaosong Huang, and Liangjun Liu. 2019. Location2vec: A situation-aware representation for visual exploration of urban locations. *IEEE Trans. Intell. Transport. Syst.* 20, 10 (2019), 3981–3990.

[158] Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*. 19–27.

Received 11 April 2023; revised 3 January 2024; accepted 12 March 2024