Using a Language Model's Perplexity for Evaluating a Trajectory's Outlierness

Zhihao Lyu
Khoury College of Computer Sciences
Northeastern University
San Jose, CA, United States
lv.zh@northeastern.edu

LeAnn Marie Mendoza
Khoury College of Computer Sciences
Northeastern University
San Jose, CA, United States
mendoza.l@northeastern.edu

Ying Shen
Khoury College of Computer Sciences
Northeastern University
San Jose, CA, United States
shen.ying@northeastern.edu

Jie Zhang
Khoury College of Computer Sciences
Northeastern University
San Jose, CA, United States
zhang.jie4@northeastern.edu

Mario Nascimento
Khoury College of Computer Sciences
Northeastern University
Vancouver, BC, Canada
m.nascimento@northeastern.edu

Michal Aibin
Khoury College of Computer Sciences
Northeastern University
Vancouver, BC, Canada
m.aibin@northeastern.edu

***Abstract*—Outlier trajectory detection is a long-standing research topic, and numerous scholars have proposed different solutions. However, most methods detect anomalies by focusing on the individual trajectories only, overlooking the relationships between outlier and inner trajectories. To address the issue of isolated predictions, we propose a novel trajectory data analysis approach that utilizes perplexity scores generated by language models as a metric to identify outliers in trajectory datasets. With the Bidirectional Encoder Representations from Transformers (BERT), our method can learn the features of trajectories through a self-attention mechanism and score the trajectories based on context. Subsequently, the scores are rapidly classified using clustering algorithms to detect outlier in trajectories. The proposed method has been tested on both real and synthetic datasets, significantly outperforming state-of-the-art methods and demonstrating high robustness. This research contributes to the field by integrating NLP with spatial-temporal data analysis, presenting new possibilities for urban planning, intelligent transportation system, and surveillance applications.**

***Index Terms*—Spatial-Temporal Data Mining, Outlier Trajectory Detection, Perplexity, Large Language Model, Deep Learning.**

I. Introduction

In the realm of spatio-temporal (ST) data, trajectories refer to a sequence of spatial and temporal locations that describe the path or movement of an object or entity over time. This data can be generated by a wide range of sources, including GPS devices, mobile phones, sensor networks, and surveillance cameras and represent the movement of various entities, such as vehicles, pedestrians, wildlife, ships, and even disease diagnosis [1]. The data typically includes time-stamped coordinates (latitude and longitude), and additional attributes like speed, direction, and altitude. The applications of trajectory data are far-reaching, ranging from enabling location-based services [2] such as navigation apps and targeted advertising to aiding businesses in engaging customers based on their real-time location and preferences.

There are a multitude of applications of trajectory data, including prediction, classification, and outlier detection [3]. Outlier detection holds substantial significance in the domain of ST applications, particularly for its crucial function in identifying anomalous occurrences. This detection technique finds applicability across a range of pragmatic contexts, and a plethora of researchers have historically approached this problem space with a variety of methods and proprietary algorithms [4]–[18].

In this research, we approached the task of ST Trajectory outlier detection as a natural language processing (NLP) task, leveraging the information generated *perplexity scores* [19] can afford us for classification. Our approach involved a three-fold process. We first developed a model to map trajectories from their latitude longitude space to a *corpus of sentences* with *unique tokens* that correspond to ST regions on a map, then implemented NLP strategies that harnessed the ability of an NLP model to *learn context*. We utilize the encoder from Transfomer [20] and pretraining task from BERT [21] to train our model and produce perplexity scores, a measure of the a models *familiarity with a given sequence of words* [19], for each trajectory. Finally, with the produced perplexity scores, we determine a clustering function that separates outlier and non-outlier trajectories.

In navigating through the multifaceted landscape of spatio-temporal (ST) trajectory outlier detection, this research embarks on a comprehensive exploration, unfolding in several critical stages. In the following Section II, we articulated the reasons for undertaking this research and explained how it addresses a particular gap in current studies. In Section III, we conducted a literature review on the definition of outliers, the current applications of language models in outlier analysis, and the utilization of perplexity in detection. In Sections IV and V, we provided a detailed exposition of our detection approach after introducing certain definitions. Section VI involved the design and implementation of a series

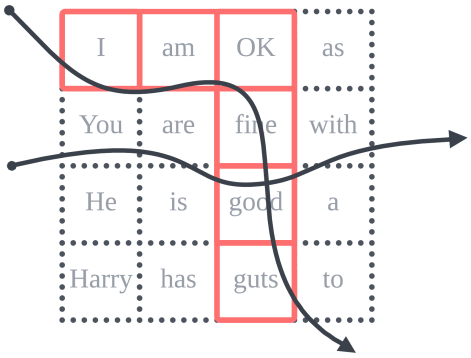


Fig. 1. Example of trajectory mapping and outlier detection

of experiments for testing. In Section VII, we presented an analysis of the experimental results. Combining the data and algorithm ablation analysis in Section VIII, we thoroughly demonstrated the superiority of our approach in predicting trajectory outliers using perplexity. Finally, in Section IX, we summarized the entire article, and in Section X, we outlined the key focus areas for future work.

II. Motivation

This research aligns with scientists' longstanding efforts to capture trajectory outliers using cutting-edge technology, with a focus on improving both accuracy and efficiency in detection. It has come to our attention that current methods based on large language models attempt to transform trajectory data, such as GPS coordinates, timestamps, and velocity, into high-dimensional features. Subsequently, these features are used to classify trajectories based on distances. However, such models are likely to lose context about the environment or other trajectories, concentrating solely on correctly converting a trajectory into a high-dimensional tensor.

The primary objective of this paper is to introduce a novel detection method that leverages the context of all trajectories within a region for anomaly detection in a single trajectory. Combining the sensitivity of language models to semantics, we transform trajectories into natural language to maximize the feature extraction capabilities provided by the model. For example, after extensive learning from a trajectory corpus, a language model may detect semantic transitions within a trajectory, where the term "good" transitions to "fine," and subsequently transforms into "guts" (Fig. 1). This process continues until the model deems the oddity of the sentence surpasses an intolerable threshold, and an outlier trajectory is detected. This predictive anomaly trajectory mechanism offers a new perspective for current trajectory modeling research, contributing to a more effective identification of outlier trajectories.

III. Literature Review

A. Trajectory Outlier Definition

Many research studies have delved into trajectory outlier detection, there is no widely accepted definition for trajectory outliers. These definitions vary based on different criteria, considering various types of motion and distinct research objectives. The most commonly used definition, as proposed by Hawkins [22], describes an outlier as an observation that significantly deviates from other observations, raising suspicions that it was generated by a different mechanism.

Some researchers employ key parameters, such as direction, density, speed, and acceleration, to identify similarities and anomalies in trajectories. For instance, Dodge et al. [23] use a time-ordered sequence of coordinates to measure movement similarity, enabling them to detect outliers. Ibrahim et al. [24] introduce speed to study the changing patterns of tropical cyclones. Johnson et al. [25] utilize acceleration sensors in smartphones to distinguish between aggressive and non-aggressive driving behaviors.

Others emphasize the role of spatial context in determining trajectory outliers. Buchin et al. [26] and Elsner et al. [27]have found that movement outliers may be influenced by underlying land/sea structures, latitude, surface temperature, and pressure. This concept also extends to vehicle trajectories, where environmental factors such as road categories and traffic conditions can influence the trajectories of vehicles.

Additionally, a trajectory outlier is defined as a deviation from high-dimensional features in the field of machine learning. Huang [28] leverages neural networks to represent trajectories as words and employs clustering functions for outlier detection. Ahmed et al. [29] and Belhadi et al. [30] utilize convolutional neural networks to transform trajectories into graphs to detect outliers. Su et al. [31] introduce meta-learning to incorporate high-dimensional features, addressing the issue of insufficient data for outlier detection.

For the purpose of our experiment, we will utilize the definition of trajectory outliers in "MiPo: How to Detect Trajectory Outliers with Tabular Outlier Detectors " [32], the corresponding paper to our data set. The definition is as follows: *an outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism* [32]. Our focus in this paper is on the detection of *detours*. Specifically, *a detour trajectory* is identified as an outlier trajectory where a sub-trajectory of the main trajectory is shifted with a random direction and distance [32].

B. Outlier Detection using Large Language Models (LLMs)

The utilization of Language Models (LLMs) for trajectory outlier detection in natural language processing plays a vital role and there are many successfully research example. Some use the dataset with time-stamps or add more feature to suitable the LLMs [33]. Some research based on GPS data for analysis with the LLMs usually create detailed Textual Representation Schema [34]. This approach typically involves reformatting the raw timestamp data into a universally recognized format like "YYYY-MM-DD HH:MM:SS". [35] [36] [37]use Latitude and longitude values which are converted into textual strings to involve labeling each coordinate with clear and distinct identifiers, such as "Lat: [latitude value],

Long: [longitude value]" in order enhance the model's ability to interpret and analyze location-specific information. Some previous research [38] use additional features provides a richer context for each GPS data point let the LLMs model to develop a more nuanced understanding of the data.

Sentences that fall into the category of outliers are often characterized by their peculiarity, unexpectedness, or substantial deviation from the usual and coherent patterns found in natural language. This can be quantified using a *perplexity score*.

C. Perplexity Scores

A language model's perplexity score is a key metric for evaluating its performance in NLP tasks. It measures how well the model predicts a word. By computing the logarithm of the cross-entropy for each word, perplexity serves as a quantifiable indicator of sentence "unlikeliness", making it useful for identifying anomalies. In previous research, perplexity has been employed as a metric for detecting anomalies across various domains. Girdhar et al. [39] utilized perplexity scores of images to detect the distribution of crabs in underwater environments. Dairi et al. [40] integrates temperature and gas sensor data for the detection of drunk driver. The perplexity is more widely applicable in NLP tasks. Jansen et al. [41] and Todd et al. [42] utilize perplexity to discern harmful content and semantic errors. Mitrović et al. [43], Fernández-Pichel et al. [44] utilize machine learning models in conjunction with perplexity to identify content generated by ChatGPT and eliminate unnecessary HTML blocks, respectively. Sun et al. [45] demonstrates the utilization of perplexity to identify anomalies in behavioral patterns. In summary, perplexity serves as a reliable metric for anomaly detection. However, it is crucial to note that other evaluation metrics, such as F1 score or AUC, often serve as important complementary measures in evaluating performance.

D. Perplexity Score Classification

In utilizing perplexity scores for classifying tasks, identifying an optimal threshold is crucial to delineate between outlier and non-outlier classifications. Perplexity scores, fundamentally statistical measures of how well a probability model predicts a sample [19], are pivotal in tasks involving language models and anomaly detection. The threshold determination hinges on balancing sensitivity and specificity, often employing statistical techniques or machine learning algorithms to ascertain the most effective demarcation [19]. A critical tool in this process is the Area Under the Receiver Operating Characteristic Curve (AUROC), which serves to maximize the accuracy of binary classifications by evaluating the trade-offs between true positive rates and false positive rates [46].

The methods for determining the threshold are vary between supervised and unsupervised training. Mitrović et al. [47] and Sun et al. [48] both use perplexity obtained after training the model on a normal training set as a baseline, and then identify anomalous sentences in the test set with perplexity greater than this baseline. In contrast, Jansen et al. [41] train the model on an data set with adult and harmful content and use the model's perplexity as a baseline, considering sentences in the test set with perplexity greater than the baseline as normal. This significantly improves the robustness of anomaly detection. With a well-annotated anomaly data set, although fine-tuning is still required based on other metrics, determining the perplexity threshold becomes relatively straightforward. However, in unsupervised learning, determining the threshold is more tedious. Todd et al. [49] first calculate a baseline perplexity using a pretrained model without any fine-tuning on in-domain knowledge. Then, they use a model fine-tuned in a specified domain to calculate perplexity on the same test set and perform anomaly detection based on the difference between the two perplexities. Fernández-Pichel et al. [44], Kim et al. [50], and Todd et al. [49], due to a lack of extensively labeled data, continuously adjust different perplexities during the experimental process and find a suitable perplexity as a threshold based on F1 score and AUC.

IV. Preliminaries and Problems Definition

In this section, we first present preliminaries of the basic concepts. Then, we give a formal problem definition.

A. Preliminaries

Trajectory Point: A trajectory point *p* is a tuple (*x, y*) that contains the latitude *x* and longitude *y* of its current position.

$$p = (x, y)$$

Trajectory: A trajectory $\mathcal{T}$ is a set that contains at two *p*.

$$\mathcal{T} = (p_1, p_2, p_3, ..., p_n)$$

Trajectory Sentence: A trajectory sentence $\mathcal{T}_e$ is a set contains natural language words *e* mapped from trajectory $\mathcal{T}$.

$$\mathcal{T}_e = (e_1, e_2, e_3, ..., e_n)$$

Trajectory Dataset: A trajectory dataset $\mathcal{D}_\mathcal{T}$ is a set that contains at least two $\mathcal{T}$.

$$\mathcal{D}_\mathcal{T} = (\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, ..., \mathcal{T}_n)$$

Geographical Boundary: A geographical boundary $\mathcal{G}$ is a square region formed by the southernmost, northernmost, westernmost, and easternmost *p* among all trajectory dataset $\mathcal{D}_\mathcal{T}$.

Language Model: A language model Ψ can be obtained by employing a well-designed training method β on an unlabeled dataset $\mathcal{D}_\mathcal{T}$.

$$\beta : \mathcal{D}_\mathcal{T} \rightarrow \Psi$$

Perplexity Score: A perplexity score *ppl* is the metric for classifying a trajectory $\mathcal{T}$. It can be calculated using Ψ, $\mathcal{T}$ and an aggregation function g for calculating average trajectory perplexity.

$$\Psi : \{\mathcal{T}, g\} \rightarrow ppl$$

Clustering Function: A clustering function $\mathcal{K}$ can categorize *ppl* into two classes with $label_T$.

$$\mathcal{K} : ppl \rightarrow label_T$$

$$label_T = \begin{cases} 1 & \text{Inner} \\ 0 & \text{Outlier} \end{cases}$$

B. Problem Definition

The problem of identifying outlier trajectories can be formalized as follows: leveraging a dataset $\mathcal{D}_\mathcal{T}$ comprising unlabeled trajectories, we utilize the model Ψ and the functions g and $\mathcal{K}$ to conduct binary classification on the perplexity scores *ppl* of unseen trajectories.

V. Methodology

In this section, we introduce a trajectory detection method based on perplexity. Inspired by the tremendous success of BERT pretrained models in NLP tasks, we decided to leverage a BERT-style pretraining task and the encoder in the Transformer as the backbone of our model Ψ. We trained our model on trajectory dataset unsupervisedly and calculated the perplexity of sentences.Then we employ the K-means algorithm as our clustering function $\mathcal{K}$ to classify the perplexity of each sentence. Fig. 2 illustrates the overall architecture of our approach. The methodological details are organized into four subsections.

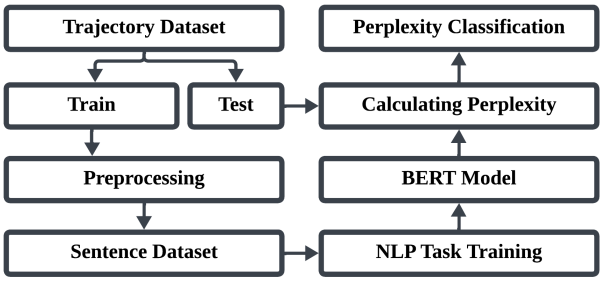


Fig. 2. Approach Overview

- **Trajectory Preprocessing**, defining the procedure of dividing the map into different areas and transforming the raw trajectory into natural language sentences;
- **Model Architecture**, describing how the trajectory is processed by the deep learning model, allowing the model to learn the underlying features of hidden patterns;
- **Perplexity Calculation**, utilizing the pretrained model to predict each word in sentences, constructing mapping relationships between perplexities and trajectory.
- **Trajectory Classification**, employing the K-means clustering function to categorize perplexity into two groups, representing inner trajectories and outliers.

A. Trajectory Preprocessing

Encoding trajectory based on spatio-temporal data into natural language systems is the first challenge to address. The mapping function ϕ needs to map potentially infinite locations to a finite vocabulary, and the similarity between two words should reflect their spatial proximity, i.e., their frequency and semantic meaning should be similar:

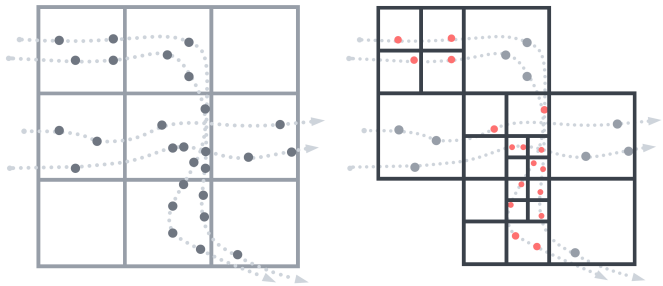
$$\phi : T_n \to \{e_1, e_2, ..., e_n\}$$

where n is the number of p in $\mathcal{T}$, and e represent an natural language word from mapped trajectory sentence $\mathcal{T}_e$.

We propose a grid-based mapping function based on the geographical boundary $\mathcal{G}$. This grid is tailored to maintain spatial uniformity, ensuring the region is evenly divided by small cells. However, merely defining a grid cannot capture the inherent nuances in real-world trajectory data, which often exhibit dense or clustered patterns. We also employ a density-based function ϱ. By evaluating the density ρ of trajectory points p within a grid cell, ϱ utilizes a quadtree for hierarchical recursively remapping (cf. fig. 3), this step ensures that our trajectory mapping maintains consistent spatial resolution.

$$\varrho\left(e_o, \mathcal{G}\right) = \begin{cases} e_r & \text{if } \rho > \gamma \\ e_o & \text{else} \end{cases}$$

where e_o is original mapped natural language word, e_r is the remapped natural language word, γ is a threshold obtained through experiments.



(a) Before remapping (b) After remapping

Fig. 3. In (a), when cells are generated on a fixed length, too many points share the same mapping. In(b), cells with higher density (4 in this figure) are remapped and empty cells are eliminated.

Finally, we introduce the drafting parameter α and the missing parameter β for data augmentation function ω to generate drafting and missing trajectory sentence $\mathcal{T}_d$, $\mathcal{T}_m$. By artificially introducing noise, we enhance the adaptability of our model.

$$\omega : \{\alpha, \beta, \mathcal{T}_e\} \to \mathcal{T}_d, \mathcal{T}_m$$

Algorithm 1 demonstrates the steps of the preprocessing trajectory dataset. The input includes $\mathcal{G}$, $\mathcal{D}_\mathcal{T}$, ϱ, α, β, ϕ, the output is a mapped trajectory sentence dataset $\mathcal{D}_E$

B. Model Architecture

We built our model based on the state-of-the-art Transformer architecture. While we used the same structure, our model is trained from scratch on trajectory sentences. As we don't need the model to directly identify outlier but rather to learn the relationships between trajectories, inspired by the tremendous success of BERT pretraining tasks, we adopt two similar training tasks: masked word prediction(MWP) and next sentence prediction(NSP). In the MWP, we replace 12% of the words with "[MASK]" and 1.5% of the words with other words. In the NSP task, we split a trajectory sentence into

Algorithm 1: Trajectory Preprocessing

Input: $\mathcal{G}$, $\mathcal{D}_{\mathcal{T}}$, ϱ, α, β, ϕ
Data: $\mathcal{D}_E \leftarrow \emptyset$
Output: $\mathcal{D}_E$
for $\mathcal{T} \in \mathcal{D}_{\mathcal{T}}$ **do**
  $\mathcal{T}_e \leftarrow \phi(\mathcal{T})$ // *trajectory to sentence*
  for $e_o \in \mathcal{T}_e$ **do**
    $e_o \leftarrow \varrho(e_o, \mathcal{G})$ // *hierarchical remapping*
  end
  $\mathcal{D}_E \cup \{\mathcal{T}_e\}$
  $\mathcal{T}_d, \mathcal{T}_m \leftarrow \omega(\alpha, \beta, \mathcal{T}_e)$ // *data augmentation*
  $\mathcal{D}_E \cup \{\mathcal{T}_d\}$
  $\mathcal{D}_E \cup \{\mathcal{T}_m\}$
end
return $\mathcal{D}_E$

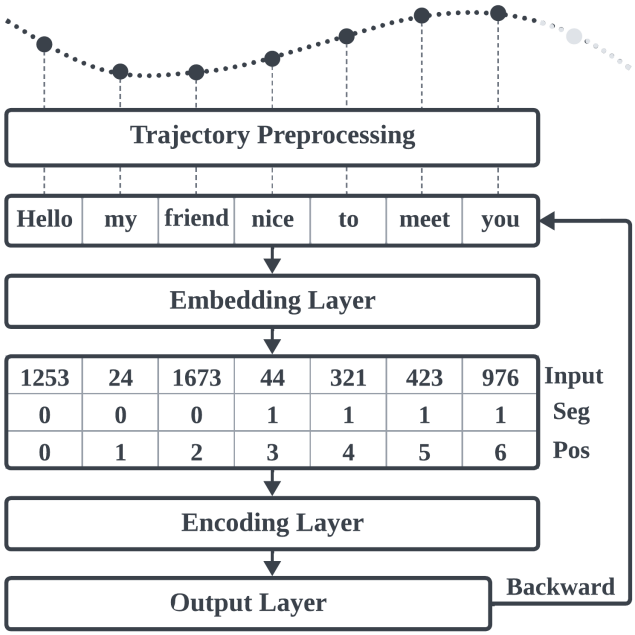


Fig. 4. Model Architecture Overview

two sub-sentences and ask the model to predict the positional relationship between these two sentences. To ensure the model successfully completes these two tasks, we divide the model into three parts during construction: the embedding layer, the encoding layer, and the output layer. Fig. 4 illustrates an overview of our model architecture.

1) Embedding layer: In the embedding layer, we map trajectory sentences to numerical IDs with special tokens "[PAD]", "[MASK]", "[CLS]", and "[SEP]" to built our own vocabulary and add adequate separation to the sentences. Finally, a processed sentence is transformed into a learnable embedding E_s with *d*-dimension features *fe* as follow:

$$E_s = \{fe_1, fe_2, fe_3, ..., fe_d\}$$

In addition, we also transform the timestamp and absolute positional relationship using the same method into embeddings E_t and E_p. Finally, the three embeddings are added together to form a learnable final embedding E.

$$E = E_s + E_t + E_p$$

2) Encoding layer: Our encoding layer is composed of transformer encoder blocks and a features pooler. The encoder leverages multiple attention heads h to learn bi-directional spatial-temporal patterns from the entire trajectory. We utilize the following equations to calculate the attention:

$$attn_T = \|_{i=0}^{h} softmax(\frac{\mathrm{Q}_i \cdot \mathrm{K}_i^T}{\sqrt{d/h}}) \cdot \mathrm{V}_i$$

where $\|$ stands for the concatenation operator, h is the number of attention heads, i is the head index. Q, K, and K represent three matrices obtained through linear transformations of embeddings E, and K^T is the transpose of matrix K. Finally, the layer normalization function $LayerNorm$, a feed-forward network FC and a features pooler $Pooler$ are applied to generate the hidden tensors as:

$$ln = LayerNorm(attn_T)$$

$$\mathcal{Z} = LayerNorm(FC(Relu(FC(ln))))$$

$$\mathcal{Z}_n = Pooler(\mathcal{Z})$$

where ln refers to the output of the layer normalization function. $\mathcal{Z}$ is the final output representing the learned tensors from a Transformer encoder. $\mathcal{Z}_n$ is the tensors that after feature extraction by the pooler and are used in the NSP task. Our model comprises multiple encoder blocks, and the greater the number of encoders, the stronger the learning capability of the model.

3) Output layer: The model's output layer comprises a projection layer and a training-task orientated loss function. The projection layer is responsible for transforming the tensor from the model's encoding layer into the required dimensions for the task. In our model, the projection layer $\mathcal{P}$ transforms $\mathcal{Z}$ into $\mathcal{Z}_v$ according to the size of the vocabulary for word prediction and $\mathcal{Z}_n$ into a 2-dimensional tensor $\mathcal{Z}_b$ for binary classification.

$$\mathcal{P} : \{\mathcal{Z}, \mathcal{Z}_n\} \rightarrow \{\mathcal{Z}_v, \mathcal{Z}_b\}$$

We adopt Cross Entropy as the loss function for our model. Since the MWP task is essentially a multi-classification task with the number of categories equal to the total number of words in the vocabulary, cross-entropy, coupled with the softmax function, can better predict different categories. The cross entropy loss $\mathcal{L}$ for N data points can be calculated as follows:

$$\mathcal{L}_{mwp} = -\frac{1}{N}\sum_{i}\sum_{c=1}^{M} y_{ic} log(p_{ic})$$

where M is the vocabulary size. y_{ic} is the sign function. If the class of sample i is equal to c, it takes the value 1; otherwise,

Algorithm 2: Model Training

Input: $\mathcal{D}_E, \Psi, \mathcal{L}_{mwp}, \mathcal{L}_{nsp}$
Data: lr, $epochs$, $ratio$, $validation$
Output: Ψ_{best}
$epoch \leftarrow 0$
$opt \leftarrow optimizer(\Psi, lr)$
$split\ \mathcal{D}_E\ into\ \mathcal{D}_{train}, \mathcal{D}_{test}\ by\ ratio$
while $epoch < epochs$ **do**
  for $\mathcal{T}_e \in \mathcal{D}_{train}$ **do**
    $\mathcal{Z}_v, \mathcal{Z}_b \leftarrow \Psi(\mathcal{T}_e)$
    $\mathcal{L} \leftarrow \mathcal{L}_{mwp}(\mathcal{Z}_v) + \mathcal{L}_{nsp}(\mathcal{Z}_b)$
  end
  $opt.zero\_grad()$ // $zeroing\ out\ gradients$
  $\mathcal{L}.backward()$ // $accumulating\ gradients$
  $opt.step()$
  $\Psi_{best} \leftarrow validation(\Psi)$
  $epoch \leftarrow epoch + 1$
end
return Ψ_{best}

Algorithm 3: Perplexity Calculation

Input: $\mathcal{D}_E$, Ψ_{best}, $\mathcal{L}_{mwp}$, $\mathcal{L}_{nsp}$
Data: $ratio$, $validation$
Output: $\mathcal{D}_P$
$split\ \mathcal{D}_E\ into\ \mathcal{D}_{train}, \mathcal{D}_{test}\ by\ ratio$
for $\mathcal{T}_e \in \mathcal{D}_{test}$ **do**
  $list \leftarrow \emptyset$
  $N \leftarrow |\mathcal{T}_e|$
  for $e \in \mathcal{T}_e$ **do**
    $e \leftarrow$ "[MASK]" // $mask\ a\ word$
    $\mathcal{Z}_v, \mathcal{Z}_b \leftarrow \Psi_{best}(\mathcal{T}_e)$
    $\mathcal{L} \leftarrow \mathcal{L}_{mwp}(\mathcal{Z}_v) + \mathcal{L}_{nsp}(\mathcal{Z}_b)$ // $probability$
    $ppl \leftarrow e^{\frac{\mathcal{L}}{N}}$
    $list \cup \{ppl\}$
  end
  $avg \leftarrow \overline{list}$ // $get\ average\ perplexity$
  $\mathcal{D}_P \cup \{avg\}$
end
return $\mathcal{D}_P$

it takes the value 0. p_ic represent the probability of sample i is predicted as class c.

Additionally, since we later need to compute perplexity using the model, and perplexity calculation is based on the cross-entropy loss. Employ it as the loss function helps the model adapt to the environment in the training data, resulting in higher perplexity for outlier trajectories in the test set.

As for the binary classification task of NSP, it's actually a special form of multi-classification. It's loss $\mathcal{L}_{nsp}$ for N data points can be computed by:

$$\mathcal{L}_{nsp} = \frac{1}{N} \sum_{i} -[y_i \cdot log(p_i) + (1 - y_i) \cdot log(1 - p_i)]$$

where y_i is the label of sample i, where positive class is 1 and negative class is 0; p_i represents the probability that sample i is predicted as the positive class. In the end, we add these two losses together to form our model's hybrid loss function $\mathcal{L}$, which is utilized for training the model:

$$\mathcal{L} = \mathcal{L}_{mwp} + \mathcal{L}_{nsp}$$

Algorithm 2 outlines the process of training our model. Given a dataset of trajectory sentences $\mathcal{D}_E$, an initial model Ψ, and two loss function $\mathcal{L}_{mwp}$, $\mathcal{L}_{nsp}$. The output is the model Ψ_{best} performs best after training.

C. Perplexity Calculation

Due to the fact that our model does not generate words like other language model like LSTM or N-gram, sequentially from left to right, but instead employs self-attention mechanism to focus on all words in the sentence regardless of their order, the computation of its perplexity is slightly different. Using the idea from Miaschi et al. [51], we mask each word in the sentence one by one, calculate the probability of each word, and then take the average as the probability of the entire sentence. The probability p of a $\mathcal{T}_e$ is illustrated as below:

$$p(\mathcal{T}_e) \approx \prod_{i=0}^{k} p(w_i \mid context)$$

Where *context* represents the all information both before and after the masked word in $\mathcal{T}_e$, k is the number of natural language words in $\mathcal{T}_e$. Afterward, we can use the probability of a sentence to calculate its perplexity. The final representation of perplexity ppl for $\mathcal{T}_e$ is as follows:

$$ppl_{\mathcal{T}_e} = e^{\frac{p(\mathcal{T}_e)}{N}}$$

The *N* represents the total length of the sentence $\mathcal{T}_e$ and e is the Euler's number in above equation. Algorithm 3 outlines the procedure for calculating perplexity. The input are trajectory sentences $\mathcal{D}_E$, trained model Ψ_{best} and loss function $\mathcal{L}$. The output is a set of perplexity $\mathcal{D}_P$.

D. Trajectory Classification

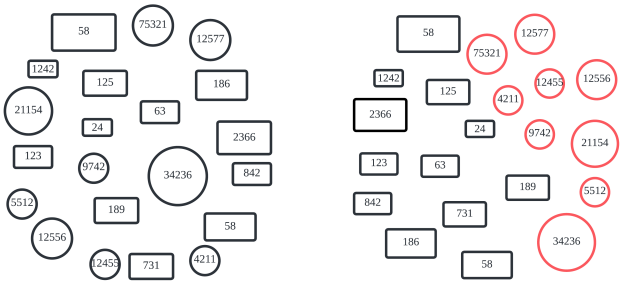
The inference phase involves both generation and classification tasks, serving the purpose of assessing the model's performance after training. The fundamental concept is using unseen trajectories to calculate perplexity with the method mentioned above and classify it as outlier or inner(cf. Fig. 5).

In our implementation, we use K-means as clustering function $\mathcal{K}$ to perform binary classification. Because it allows for unsupervised training after providing the number of categories, aligning with the objectives of our research.

$$\mathcal{K} : ppl \rightarrow label_T$$

$$label_T = \begin{cases} 1 & \text{Inner} \\ 0 & \text{Outlier} \end{cases}$$

K-means function divides a set of *N* samples *X* into *K* disjoint clusters *C*, each described by the mean μ_j of the samples in the cluster. The means are commonly called the



(a) Before Classification (b) After Classification

Fig. 5. There are many perplexity scores range from 50 to 10000 in circles and rectangles in (a). After classification using clustering function in(b), scores are categorized as two groups representing inner and outlier.

cluster centroids $\mathcal{C}d$; note that they are not, in general, points from X, although they live in the same space. The K-means algorithm aims to choose centroids that minimise the inertia, a measure of how internally coherent clusters are, or within-cluster sum-of-squares criterion:

$$\mathcal{C}d = \sum_{i=0}^{n} \min_{\mu_j \in \mathcal{C}} (\| x_i - \mu_j \|^2)$$

Despite our trajectoies training is unsupervised, given the small proportion of abnormal trajectories in the dataset, the model is highly sensitive to outliers. This sensitivity leads to notably higher perplexity scores compared with inner trajectories. The significant difference between outlier and inner dramatically helps in detecting tasks, which is confirmed by the following section.

VI. Experiments

A. Experimental Setup

For our experiments, we utilized an Ubuntu Focal 20.04.5 LTS server equipped with an AMD Ryzen 9 5950X 16-Core Processor boasting 32 CPUs, 64GB of RAM, and a 24GB NVIDIA RTX 4090 GPU. Our model was constructed using Python, implementing each component of the Transformer in the PyTorch framework to achieve training for BERT-style tasks. PyTorch Lightning Framework was employed to compute evaluation metrics.

B. Dataset

We conducted experiments using one real-world trajectory dataset and 9 synthetic datasets. The unselected dataset consisted of 1,710,670 trajectories, involving 442 taxis in Porto, Portugal [52], covering the period from 7 January 2013 to 30 June 2014, with an average GPS sampling interval of 15 seconds. From this dataset, we selected a subset consisting of 1,003 inner trajectories and 112 outlier trajectories as our original dataset, as they were labeled by Yang et al. [32] and provided a more manageable evaluation of performance.

We divided the real-word (RW) dataset into 70% for training, 30% for testing. We chose a larger amount of data than normal for testing for two reasons. Firstly, we can utilize data augmentation to increase the quantity of training data. Secondly, a sufficient amount of data is needed for improved binary classification based on perplexity in test phase. Our synthetic datasets were generated based on the training data. We manually introduced drifting and missing coordinates (drafting distance ranging from 80 feet to 130 feet) to assess the model's performance under extreme conditions. Depending on different objectives, we categorized synthetic dataset into three types: synthetic drifting (SD), synthetic missing (SM), and synthetic drifting and missing (SDM). Also, we employed varying drift and missing rates to create our synthetic datasets. For instance, the 0.1D_0.2M dataset indicates that 10% of trajectories have drifting, while 20% of trajectories have missing coordinates. In the case of 0_0.4M dataset, it implies that none of the trajectories have drifting, but 40% of them have missing coordinates. Further details are provided in both Table 1 and Table 2.

TABLE 1
Composition of Trajectory Dataset

Type	Name	Word	Coordinate
RW	Train	4077	137841
	Test	1874	16612
	Original	1871	16542
SD	0.1D_0M	2873	37413
	0.2D_0M	3210	42445
	0.4D_0M	3591	50821
SM	0D_0.1M	2873	37413
	0D_0.2M	2513	37169
	0D_0.4M	2627	40993
SDM	0.1D_0.1M	2893	40061
	0.2D_0.2M	3156	47429
	0.4D_0.4M	3636	63965

TABLE 2
Trajectory Length and Distribution

Type	Name	Trajectory Length (min, max, mean)	Inner	Outlier
RW	Train	19, 97, 47	2589	297
	Test	36, 75, 49	301	34
	Original	36, 75, 49	300	33
SD	0.1D_0M	37, 88, 50	661	73
	0.2D_0M	36, 91, 52	722	80
	0.4D_0M	37, 95, 54	842	93
SM	0D_0.1M	21, 73, 47	661	73
	0D_0.2M	20, 74, 46	722	80
	0D_0.4M	20, 75, 43	842	93
SDM	0.1D_0.1M	23, 94, 49	727	81
	0.2D_0.2M	21, 97, 49	866	96
	0.4D_0.4M	21, 97, 48	1179	131

C. Baseline and Evaluation Metrics

We selected (1) MiPo [32], one of the state-of-the-art non-deep-learning methods for outlier trajectory detection, and (2) GADFormer [53], one of the state-of-the-art deep-learning methods, as control groups to compare with our model on same datasets. To explore the performance of our model, due to the dataset's imbalance, we employed the F1 score as the evaluation metric. The F1 score F_1 is a value between 0 and 1, with higher values indicating better performance. The F1 score can be calculated by:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

D. Implementation Details

We divided the experimental area into a 256 x 256 grid, with each grid representing 100 - 130 feet in the real world. We used 0.3 and 0.3 as drafting and missing parameters respectively to augment the training data. We then used the mapping method described in Section V.A to map 137,841 coordinates from the 2,886 trajectories into natural language, resulting in a dictionary of size 4,077. We then constructed tasks using the method described in the Devlin et al.'s work [21] and continuously adjusted hyperparameters based on experimental results. The final settings were a hidden size of 768, a batch size of 768, a 12-layer decoders, and 12 self-attention heads. The learning rate started at 1e-4 and was reduced to a minimum learning rate of 1e-6 after 100 epochs using a cosine annealing scheduler. We then used the model to calculate the perplexity of all trajectories in 30% of the data. Bisect K-means algorithm is used to classify perplexity into two categories and evaluated the results on the test set.

VII. Results and Analysis

A. Compared with MiPo

We first compared our model with MiPo. Although MiPo's F1 score is competitive in original datasets, it is still about ten percentage lower than our model. When we introduced draft and missing data, MiPo significantly reduced its F1 score(cf. Table 3, 4, 5, 6).

According to experimental results, both MiPo and our model consistently experience accuracy drop as the drifting parameter decreases from 0.1 to 0.4, highlighting the significant impact of drift for both approaches(cf. Table 5). Traditional models like MiPo, utilizing K-means for distance comparison, are more sensitive to trajectory drifting, as they may identify drifting coordinates as outliers. In contrast, our model, which fundamentally does not require the calculation of physical distances between trajectory points and has extensive pretraining using augmented data, naturally adapts well to extreme scenarios in the test dataset. This is the reason why our model performs well even under the influence of drifting.

Concerning synthetic missing dataset, the F1 score of the MiPo significantly drops, while our model maintains a high level of performance(cf. Table 4). However, this may be attributed to the fact that the coordinates missing we applied is distributed evenly across the entire trajectory, resulting in minimal alteration to the overall trajectory trend. In future work, we should explore group coordinates missing over a specific duration to further improve our model.

TABLE 3
Results on Original Dataset

	Original
Ours	0.923
MiPo	0.830
GADFormer	0.843

TABLE 4
Results on SM Dataset

	0D_0.1M	0D_0.2M	0D_0.4M
Ours	0.891	0.911	0.881
MiPo	0.693	0.672	0.634
GADFormer	0.796	0.829	0.840

TABLE 5
Results on SD Dataset

	0.1D_0M	0.2D_0M	0.4D_0M
Ours	0.921	0.925	0.914
MiPo	0.703	0.710	0.692
GADFormer	0.807	0.784	0.737

TABLE 6
Results on SDM Dataset

	0.1D_0.1M	0.2D_0.2M	0.4D_0.4M
Ours	0.901	0.911	0.880
MiPo	0.680	0.634	0.625
GADFormer	0.799	0.801	0.794

B. Compared with GADFormer

Compared our model with GADFormer, which also uses deep learning methods and uses a Transformer encoder as a feature extractor detect outlier trajectory. Our model dominated all test datasets again(cf. Table 3, 4, 5, 6). This is mainly due to our different training methods. The unsupervised training in GADFormer is based on an assumption that anomalies in trajectories are rare and will not significantly impact the model. In fact, this doesn't qualify as true unsupervised training. However, we do not make any assumptions about the data; instead, our MWP and NSP training task are completely

unsupervised, which means we can objectively learn patterns in the dataset without being limited by the normal and abnormal data ratio in datasetis. This is why we performed better in all datasets.

It's worth mentioning that GADFormer's F1 score increases as more data is missing in SM datasets(cf. Table 4 and Table 6). We believe this is because the GADFormer is designed for shorter trajectories, such as walking or cycling. Also, because coordinates are evenly missing, shortening the trajectory length improves model's performance. This principle can also explain why its accuracy in the 0.1D_0.1M and 0.4D_0.4M datasets is similar.

VIII. Ablation Analysis

A. Data augmentation

The overall data results indicate that our model not only predicts more accurately in the original dataset but also performs exceptionally well on synthetic datasets with introduced drafting and missing data. However, as we trained the model(augmented model) using an augmented dataset, we want to explore whether the high accuracy is attributed to perplexity-based detection approach or data augmentation. Therefore, keeping other parameters and methods constant, we trained the model using only the original data(original model) and tested its performance on all datasets. Our un-augment datasets for training are shown in Table 7. All test result are shown as Table 8, 9, 10, where Aug means the results for augmented model, No_Aug means that of original model.

The original model's F1 score decreased compared to the augmented model, but it's still about 5% higher than the prediction results of the GADFormer model. This not only demonstrates the necessity of data augmentation but also underscores the superiority of perplexity-based detection over other deep learning models. Although the draft and missing ratio(0.3, 0.3 respectively) we used in the data augmentation are different with the test set, our enhanced model performed well on different test dataset. This indicates that data augmentation indeed enables the model to learn the relative relationship between the noise and the trajectory itself, rather than simply making predictions through rote memorization. Furthermore, we found that the model's performance decreased most significantly in SD datasets and SDM datasets, while there was no significant decline in SM datasets. We believe this is because the model's recognition of spatial relationships plays a role. It can identify the relative position of each coordinate in the trajectory, thereby determining outlier based on those relationships. In the future, we can explore the possibility of few shots training by reducing the number of coordinates in trajectories and the total number of trajectories.

TABLE 7
Un-augment Dataset

	Train	Test
Word	2877	1871
Coordinate	34879	16542
Trajectory Length (min, max, mean)	36, 74, 49	36, 75, 49
inner	702	300
outlier	78	33

TABLE 8
Original Model Results On SM & Original Dataset

	0D_0.1M	0D_0.2M	0D_0.4M	Original
Aug	0.891	0.911	0.881	0.923
No_Aug	0.874	0.887	0.874	0.923
GADFormer	0.796	0.829	0.840	0.843

TABLE 9
Original Model Results On SD Dataset

	0.1D_0M	0.2D_0M	0.4D_0M
Aug	0.921	0.925	0.914
No_Aug	0.846	0.823	0.782
GADFormer	0.807	0.784	0.737

TABLE 10
Original Model Results On SDM Dataset

	0.1D_0.1M	0.2D_0.2M	0.4D_0.4M
Aug	0.901	0.911	0.880
No_Aug	0.824	0.819	0.752
GADFormer	0.799	0.801	0.794

B. Clustering algorithms

In the process of outlier detection, an important step is to perform binary classification on perplexity using clustering algorithms. We chose K-means (KM) because it is a classical algorithm that supports unsupervised training. However, we are still interested in exploring the impact of different clustering algorithms on prediction results. Keeping other variables unchanged, we replaced KM with Gaussian Mixture Model (GMM) and Hierarchical Clustering (HC) to classify perplexity, because all three methods support unsupervised training, aligning with the research goals of our research. Besides of F1-score, we also recorded the average running time for finishing all tests on all datasets, aiming to investigate their efficiency. The result across all datasets is presented in Table 11, 12, 13, 14. "A" in column represents the algorithm's performance on the perplexity generated by the augmented model, "O" indicates that of original model, "K" represents K-means, "H" represents Hierarchical Clustering, and "G" represents Gaussian Mixture Model. For example, "KO" denotes outcomes derived from the original model utilizing K-means as the clustering algorithm; "GA" represents results obtained from the augmented model with Gaussian Mixture Model employed as the clustering algorithm.

1) Gaussian Mixture Model: By comparing KM and GMM, we observed that although their runtimes are similar, but

TABLE 11
RESULTS ON SM DATASET

	0D_0.1M	0D_0.2M	0D_0.4M	Original
KA	0.891	0.911	0.881	0.923
KO	0.874	0.887	0.874	0.923
GA	0.640	0.613	0.616	0.610
GO	0.523	0.507	0.567	0.926
HA	0.901	0.906	0.901	0.923
HO	0.888	0.889	0.904	0.926

TABLE 12
RESULTS ON SD DATASET

	0.1D_0M	0.2D_0M	0.4D_0M
KA	0.921	0.925	0.914
KO	0.846	0.823	0.782
GA	0.637	0.606	0.606
GO	0.512	0.452	0.469
HA	0.922	0.904	0.906
HO	0.858	0.774	0.815

TABLE 13
RESULTS ON SDM DATASET

	0.1D_0.1M	0.2D_0.2M	0.4D_0.4M
KA	0.901	0.911	0.880
KO	0.824	0.819	0.752
GA	0.616	0.582	0.571
GO	0.500	0.491	0.916
HA	0.899	0.901	0.914
HO	0.817	0.832	0.915

TABLE 14
RESULTS ON RUNNING TIME COMPARISON

Algorithm	KA	KO	GA	GO	HA	HO
Time(s)	0.687	0.659	0.686	0.672	4.995	5.205

the classification accuracy of GMM significantly decreases, whether in augmented model or original model. We attribute this to the distribution of perplexity values. While both GMM and KM utilize designated centers for cluster partitioning, GMM assumes that the data within a cluster follows a Gaussian distribution. However, although the perplexity scores calculated by the model exhibit a significant difference between outlier and inner data, the value of scores in their own category is randomly distributed. Therefore, using probabilities derived from various Gaussian distributions, primarily employed by the GMM, to classify becomes inaccurate. In contrast, KM classifies based on the Euclidean distance between each value, resulting in uniformly distributed data within a clusters, aligning with the characteristics of distribution of perplexity values. Hence, KM outperforms GMM significantly in our all tests.

2) Hierarchical Clustering: Comparing the experimental results of KM and HC, we first noticed that there is not a significant difference in the accuracy of prediction. We attribute this to the underlying similarity in the classification process of HC and KM. They both use Euclidean distance between numerical values as the criterion for clustering. However, we observed that KM runs much faster than HC. By comparing details of their implementation, we found that this is due to the different time complexities of the algorithms. The HC algorithm, for the problem of dividing n elements into k classes, requires comparing the similarity between any two elements, and this process needs to be repeated $(n-k)$ times, resulting in a time complexity of $O(n^3)$. On the other hand, KM involves calculating the distance from randomly initialized k centroids to n elements and repeating this process c times. Therefore, the time complexity of KM is $O(ckn)$. Since k is only 2 in our study, and c is usually a relatively small constant, the time complexity of KM is much smaller than that of HC, explaining why it is significantly faster.

In summary, K-means excels in both accuracy and speed compared to other algorithms. However, in the future, we will continue to experiment with various clustering algorithms in different scenarios and problems to explore algorithms that perform better with different datasets.

IX. CONCLUSION

In this paper, we introduced a new method for detecting outlier trajectory using perplexity as an indicator in language models. Our goal is to treat trajectories as a type of language and use large language models' self-attention mechanism to better learn each trajectory's features, thereby achieving higher detection rates under various conditions. By training with BERT-style tasks and augmented datasets, our model achieved higher F1 score than MiPo and GADFormer not only in real-world datasets but also in extreme datasets with draft and missing coordinates. This not only proves our method's superiority over other methods but also demonstrates the necessity of introducing noise to increase trajectory diversity in training sets. Currently, language models are still being explored for trajectory analysis. In the future, we plan to continue researching language models' predictive and inferential capabilities in spatial-temporal data.

X. FUTURE WORK

(1) Test on more datasets. As we have only used a single area, single origin, and destination dataset for testing, we need to test in more scenarios, including different locations, different outlier ratios, different outlier types, and even different modes of transportation, to prove the reliability of perplexity and language models in trajectory analysis.

(2) Explore the possibility of few shots learning. As we found that the model can still efficiently identify abnormal trajectories when certain data is missing, we can explore the model's performance under small-size dataset in the future, including the number of trajectories and GPS coordinates in the trajectory.

(3) Real-time Detection: As our model utilizes perplexity for outlier detection, the calculation speed is faster than predicting a whole trajectory. Moreover, the self-attention mechanism allows the analysis of any trajectory segment without requiring the entire trajectory information. These two aspects provide the possibility of achieving real-time prediction by reducing the input data volume.

(4) Multi-Source Multi-Destination Trajectory Detection: Changes in trajectories may be influenced by other trajectories, such as a road accident leading all vehicles to take detours. If the trajectory information from different source and destinations can be utilized to comprehensively analyze the anomaly of a trajectory, it will help improve the accuracy of trajectory prediction and adaptability to uncertain environments.

(5) Utilization of Non-Trajectory Information: The acquisition of environmental information is crucial for trajectory anomaly detection, whether it be for vehicles or robotic vacuums. If we can obtain auxiliary information such as floor plans or traffic congestion, it could significantly enhance trajectory detection. This is an aspect we aim to explore in our future research.

REFERENCES

[1] A. Belhadi, Y. Djenouri, G. Srivastava, D. Djenouri, J. C.-W. Lin, and G. Fortino, "Deep learning for pedestrian collective behavior analysis in smart cities: A model of group trajectory outlier detection," *Information Fusion*, vol. 65, pp. 13–20, 2021. I

[2] Y. Li, X. Zhao, Z. Zhang, Y. Yuan, and G. Wang, "Annotating semantic tags of locations in location-based social networks," *GeoInformatica*, vol. 24, pp. 133–152, 2020. I

[3] E. O. Eldawy, E. O. Eldawy, M. Abdalla, A. Hendawi, and H. M. O. Mokhtar, "Spatio-temporal outlier detection." I

[4] C. C. Aggarwal and C. C. Aggarwal, *An introduction to outlier analysis*. Springer, 2017. I

[5] L. Zhang, Z. Hu, and G. Yang, "Trajectory outlier detection based on multi-factors," *IEICE TRANSACTIONS on Information and Systems*, vol. 97, no. 8, pp. 2170–2173, 2014. I

[6] D. Kumar, J. C. Bezdek, S. Rajasegarar, C. Leckie, and M. Palaniswami, "A visual-numeric approach to clustering and anomaly detection for trajectory data," *The Visual Computer*, vol. 33, pp. 265–281, 2017. I

[7] F. Meng, G. Yuan, S. Lv, Z. Wang, and S. Xia, "An overview on trajectory outlier detection," *Artificial Intelligence Review*, vol. 52, pp. 2437–2456, 2019. I

[8] J. D. Mazimpaka and S. Timpf, "Trajectory data mining: A review of methods and applications," *Journal of spatial information science*, vol. 2016, no. 13, pp. 61–99, 2016. I

[9] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104. I

[10] E. M. Knorr, R. T. Ng, and V. Tucakov, "Distance-based outliers: algorithms and applications," *The VLDB Journal*, vol. 8, no. 3, pp. 237–253, 2000. I

[11] J.-G. Lee, J. Han, and X. Li, "Trajectory outlier detection: A partition-and-detect framework," in *2008 IEEE 24th International Conference on Data Engineering*. IEEE, 2008, pp. 140–149. I

[12] A. Kut and D. Birant, "Spatio-temporal outlier detection in large databases," *Journal of computing and information technology*, vol. 14, no. 4, pp. 291–297, 2006. I

[13] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003. I

[14] T. Cheng and Z. Li, "A multiscale approach for spatio-temporal outlier detection," *Transactions in GIS*, vol. 10, no. 2, pp. 253–263, 2006. I

[15] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, and S. Li, "ibat: detecting anomalous taxi trajectories from gps traces," in *Proceedings of the 13th international conference on Ubiquitous computing*, 2011, pp. 99–108. I

[16] X. Li, J. Han, S. Kim, and H. Gonzalez, "Roam: Rule-and motif-based anomaly detection in massive moving object data sets," in *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 2007, pp. 273–284. I

[17] R. R. Sillito and R. B. Fisher, "Semi-supervised learning for anomalous trajectory detection." in *BMVC*, vol. 1, 2008, pp. 035–1. I

[18] J. P. Rogers, D. Barbara, and C. Domeniconi, "Detecting spatio-temporal outliers with kernels and statistical testing," in *2009 17th International Conference on Geoinformatics*. IEEE, 2009, pp. 1–6. I

[19] K. Arora and A. Rangarajan, "Contrastive entropy: A new evaluation metric for unnormalized language models," *arXiv preprint arXiv:1601.00248*, 2016. I, III-D, III-D

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017. I

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018. I, VI-D

[22] D. Hawkins, *Identification of Outliers*, ser. Monographs on applied probability and statistics. Chapman and Hall, 1980. [Online]. Available: https://books.google.com/books?id=fb0OAAAAQAAJ III-A

[23] S. Dodge, P. Laube, and R. Weibel, "Movement similarity assessment using symbolic representation of trajectories," *International Journal of Geographical Information Science*, vol. 26, no. 9, pp. 1563–1588, 2012. III-A

[24] A. Ibrahim, U. Turdukulov, and M.-J. Kraak, "Linking movement and environmental data: The need for representation," *International journal of applied earth observation and geoinformation*, vol. 45, pp. 95–105, 2016. III-A

[25] D. A. Johnson and M. M. Trivedi, "Driving style recognition using a smartphone as a sensor platform," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. Ieee, 2011, pp. 1609–1615. III-A

[26] M. Buchin, S. Dodge, and B. Speckmann, "Similarity of trajectories taking into account geographic context," *Journal of Spatial Information Science*, no. 9, pp. 101–124, 2014. III-A

[27] J. B. Elsner and A. B. Kara, *Hurricanes of the North Atlantic: Climate and society*. New York: Oxford University Press, 1999. III-A

[28] S. E. Huang, Y. Feng, and H. X. Liu, "A data-driven method for falsified vehicle trajectory identification by anomaly detection," *Transportation research part C: emerging technologies*, vol. 128, p. 103196, 2021. III-A

[29] U. Ahmed, G. Srivastava, Y. Djenouri, and J. C.-W. Lin, "Knowledge graph based trajectory outlier detection in sustainable smart cities," *Sustainable Cities and Society*, vol. 78, p. 103580, 2022. III-A

[30] A. Belhadi, Y. Djenouri, G. Srivastava, D. Djenouri, J. C.-W. Lin, and G. Fortino, "Deep learning for pedestrian collective behavior analysis in smart cities: A model of group trajectory outlier detection," *Information Fusion*, vol. 65, pp. 13–20, 2021. III-A

[31] Y. Su, D. Yao, and J. Bi, "Few-shot learning for trajectory outlier detection with only normal trajectories," in *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2023, pp. 1–8. III-A

[32] J. Yang, X. Tan, and S. Rahardja, "Mipo: How to detect trajectory outliers with tabular outlier detectors," *Remote Sensing*, vol. 14, no. 21, p. 5394, 2022. III-A, III-A, III-A, VI-B, VI-C

[33] B. Rouet-Leduc, R. Jolivet, M. Dalaison, P. A. Johnson, and C. Hulbert, "Autonomous extraction of millimeter-scale deformation in insar time series using deep learning," *Nature communications*, vol. 12, no. 1, p. 6480, 2021. III-B

[34] J. Leukel, J. Gonz'alez, and M. Riekert, "Machine learning-based failure prediction in industrial maintenance: improving performance by sliding window selection," *International Journal of Quality Reliability Management*, vol. 40, no. 6, pp. 1449–1462, 2023. III-B

[35] T. Julie, M. Sadouanouan, and T. Yaya, "A geolocation approach for tweets not explicitly georeferenced based on machine learning," in *International Symposium on Distributed Computing and Artificial Intelligence*. Springer, 2023, pp. 223–231. III-B

[36] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: State of the art, current trends and challenges," *Multimedia tools and applications*, vol. 82, no. 3, pp. 3713–3744, 2023. III-B

[37] K. Sun, Y. Zhu, P. Pan, Z. Hou, D. Wang, W. Li, and J. Song, "Geospatial data ontology: the semantic foundation of geospatial data integration and sharing," *Big Earth Data*, vol. 3, no. 3, pp. 269–296, 2019. III-B

[38] L. Zheng, D. Xia, X. Zhao, L. Tan, H. Li, L. Chen, and W. Liu, "Spatial–temporal travel pattern mining using massive taxi trajectory

data," *Physica A: Statistical Mechanics and its Applications*, vol. 501, pp. 24–41, 2018. III-B
[39] Y. Girdhar, W. Cho, M. Campbell, J. Pineda, E. Clarke, and H. Singh, "Anomaly detection in unstructured environments using bayesian nonparametric scene modeling," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2651–2656. III-C
[40] A. Dairi, F. Harrou, and Y. Sun, "Efficient driver drunk detection by sensors: A manifold learning-based anomaly detector," *IEEE Access*, vol. 10, pp. 119 001–119 012, 2022. III-C
[41] T. Jansen, Y. Tong, V. Zevallos, and P. O. Suarez, "Perplexed by quality: A perplexity-based method for adult and harmful content detection in multilingual heterogeneous web data," *arXiv preprint arXiv:2212.10440*, 2022. III-C, III-D
[42] G. Todd, C. Voss, and J. Hong, "Unsupervised anomaly detection in parole hearings using language models," in *Proceedings of the Fourth Workshop on Natural Language Processing and Computational Social Science*, 2020, pp. 66–71. III-C
[43] S. Mitrovi'c, D. Andreoletti, and O. Ayoub, "Chatgpt or human? detect and explain. explaining decisions of machine learning model for detecting short chatgpt-generated text," *arXiv preprint arXiv:2301.13852*, 2023. III-C
[44] M. Fern'andez-Pichel, M. Prada-Corral, D. E. Losada, J. C. Pichel, and P. Gamallo, "An unsupervised perplexity-based method for boilerplate removal," *Natural Language Engineering*, pp. 1–18, 2023. III-C, III-D
[45] L. Sun, X. Chen, Z. He, and L. F. Miranda-Moreno, "Routine pattern discovery and anomaly detection in individual travel behavior," *Networks and Spatial Economics*, vol. 23, no. 2, pp. 407–428, 2023. III-C
[46] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997. III-D
[47] S. Mitrovi'c, D. Andreoletti, and O. Ayoub, "Chatgpt or human? detect and explain. explaining decisions of machine learning model for detecting short chatgpt-generated text," *arXiv preprint arXiv:2301.13852*, 2023. III-D
[48] L. Sun, X. Chen, Z. He, and L. F. Miranda-Moreno, "Routine pattern discovery and anomaly detection in individual travel behavior," *Networks and Spatial Economics*, vol. 23, no. 2, pp. 407–428, 2023. III-D
[49] G. Todd, C. Voss, and J. Hong, "Unsupervised anomaly detection in parole hearings using language models," in *Proceedings of the Fourth Workshop on Natural Language Processing and Computational Social Science*, 2020, pp. 66–71. III-D, III-D
[50] S. Mac Kim, S. Wan, C. Paris, and A. Duenser, "Social media relevance filtering using perplexity-based positive-unlabelled learning," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 14, 2020, pp. 370–381. III-D
[51] A. Miaschi, D. Brunato, F. Dell'Orletta, and G. Venturi, "What makes my model perplexed? a linguistic investigation on neural language models perplexity," in *Proceedings of Deep Learning Inside Out (DeeLIO): The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, 2021, pp. 40–47. V-C
[52] "Taxi trajectory data," https://www.kaggle.com/datasets/crailtap/taxi-trajectory/data, accessed: 2023-09-18. VI-B
[53] A. Lohrer, D. Malik, and P. Kröger, "Gadformer: An attention-based model for group anomaly detection on trajectories," *arXiv preprint arXiv:2303.09841*, 2023. VI-C