A Demonstration of Kafy : An Extensible and Scalable Transformers-Based System for Trajectory Data Analysis

Youssef Hussein
husse408@umn.edu
University of Minnesota, USA

Mohamed F. Mokbel
mokbel@umn.edu
University of Minnesota, USA

ABSTRACT

This demo presents Kafy ; a full-fledged transformer-based system that supports a myriad of trajectory data analysis tasks. Kafy is an extensible system that allows users to extend it with additional transformers and/or trajectory operations. This first release of Kafy is equipped with three transformers and five trajectory operations, summarization, generation, prediction, imputation, and classification. Demo audience will experience Kafy through various scenarios that demonstrate its extensibility, internal operations, and supported trajectory operations.

PVLDB Reference Format:
Youssef Hussein and Mohamed F. Mokbel. A Demonstration of Kafy : An Extensible and Scalable Transformers-Based System for Trajectory Data Analysis. PVLDB, 19(12): 4598 - 4601, 2026.
doi:10.14778/3827998.3828075

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/joHussien/kafy2025.

1 INTRODUCTION

The widespread adoption of Internet-of-Things and location-tracking devices has enabled the collection of large amounts of trajectory data. Each trajectory is composed of a sequence of location points represented by latitude and longitude coordinates. The availability of trajectory data has enabled numerous trajectory analysis operations that are essential for a myriad applications and domains, including data-driven routing, traffic monitoring and forecasting, and understanding human mobility. Trajectory data analysis has been an active area of research over the past few decades, with numerous algorithms addressing various analysis tasks, including trajectory similarity search, summarization, imputation, generation, classification, prediction, outlier detection, among many others. Despite the large number of algorithms for each of these analysis tasks, there is still a lack of full-fledged trajectory analysis systems that efficiently support a wide range of trajectory analysis tasks. The reason is that each of these algorithms introduces its own new methods and data structures tailored to its specific task. It is then impractical to build a unified system when trajectory analysis procedures differ significantly. Existing trajectory systems (e.g., [3, 5, 7, 10, 13]) are limited to basic data management operations such as storage, indexing, and retrieval queries, and do not address trajectory analysis.

This demo presents Kafy ; a full-fledged extensible system that supports a myriad of trajectory data analysis tasks. Kafy leverages the recently developed transformer architectures [12] that are trained to produce (Large) Language Models, then fine tuned to support various natural language processing tasks. Kafy thinks of trajectories as yet another language. So, instead of training a transformer architecture with a (spoken) language to produce (language) models, Kafy will train it with the (unspoken) trajectory language to produce (trajectory) models. Similar to language models, trajectory models can then be fine-tuned with minimal overhead to support various trajectory analysis tasks. This idea is grounded by the fact that we can think of trajectories as language statements. A statement is composed of an ordered set of words drawn from a language dictionary, while a trajectory is composed of an ordered set of GPS points drawn from the underlying space. Statements follow rules imposed by the language, while trajectories follow constraints imposed by the road network. Words in a statement should be semantically related, while points in a trajectory should be spatially and temporally related. In terms of operations, trajectory imputation is analogous to finding a missing word in a sentence, trajectory prediction is analogous to predicting the next few words, and trajectory summarization, generation, and classification correspond to text summarization, generation, and classification.

Kafy is an extensible system, where it first provides the necessary infrastructure and basic algorithms that are needed to employ a transformer architecture to support a myriad of trajectory analysis tasks. Then, Kafy users can extend it in two orthogonal directions either by adding more transformers to support various trajectory operations and/or by adding more trajectory operations that can entertain its already trained trajectory models. This demo lets conference attendees experience Kafy in action. Participants will interact with the first release of Kafy , which comes equipped with five trajectory operations, namely, summarization, generation, prediction, imputation, and classification, and three transformers, namely, BERT [4], GPT [8], and T5 [9]. Attendees will be able to navigate through Kafy and see how to extend it with more trajectory operations and transformers, perform various trajectory operations with its web-based GUI, and upload new trajectory data to see how it is stored and used to build new models.

2 Kafy EXTENSIBLE SYSTEM ARCHITECTURE

Figure 1 shows Kafy extensible system architecture where the input is either a training dataset or a trajectory analysis request. As Kafy employs the transformer architecture, it needs to convert

This work is supported by NSF grants IIS-2203553 and OAC-2118285.

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828075

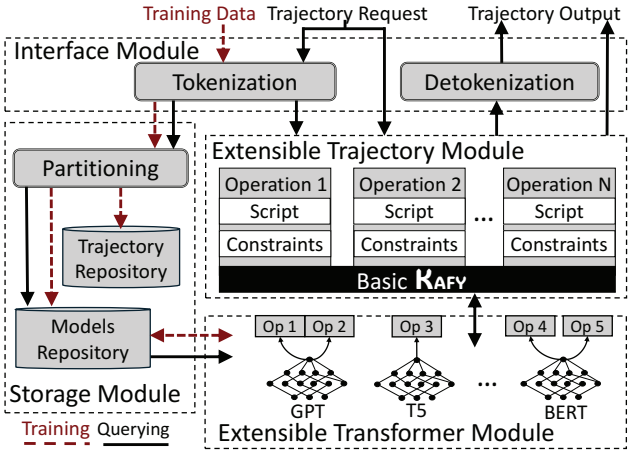

Figure 1: KAFY Architecture

the input trajectory *points* to *tokens*, which is the input unit that transformers deal with. Similarly, the output of transformers is also in terms of *tokens*, which need to be converted back to trajectory *points*. Hence, KAFY is equipped with *tokenization* and *detokenization* that employ Hexagonal partitioning [2] to represent each point by a hexagonal token. To support trajectory analysis across wide and geographically disparate areas, KAFY employs two storage structures, a *trajectory repository* and a *model repository* to store tokenized trajectories and built models for different geographical areas. Both storage structures are indexed via a *partitioning* module that employs a hierarchical pyramid structure [1]. Then, KAFY has the following two extensible modules:

The Extensible Transformer Module. This extensible module allows KAFY users to plug-in new transformers and train them on existing or new data for all supported trajectory operations. The input to this plug-in is either the training input data or the model to support the requested trajectory analysis task. It is called from the extensible trajectory plug-in to support trajectory analysis tasks.

The Extensible Trajectory Module. This extensible module allows KAFY users to plug-in new trajectory operations that are not yet supported or new algorithms for existing operations. The module is equipped with a basic KAFY algorithm that takes a trajectory analysis task and calls a specific transformer to provide an answer. To add a new operation, a user needs to define: (a) a set of *spatial constraints* that ensure the spatial validity of the output, and (b) a *script* for the operation which can either invoke the basic KAFY algorithm or override it.

3 EXTENSIBILITY IN TRANSFORMERS

There is already a plethora of transformers architectures [11] that can be categorized into three categories: *encoders*, *decoders*, and *encoder-decoders*. Similar to language tasks, some trajectory analysis tasks would be better supported by a specific category of transformers. For example, *encoder-only* transformers are best suited for tasks that require understanding the full trajectory, e.g., trajectory imputation, similarity, and clustering. *Decoder-only* transformers are more suitable for tasks that only need to understand prior trajectory points, e.g., trajectory prediction and generation. *Encoder-decoder* transformers are best suited for trajectory summarization and cleaning, which require not only understanding the full trajectory context, but also the ability to generate trajectory output. With the variety of transformers and the need to support different trajectory operations, KAFY is designed to be extensible, allowing new transformers to be added via the ADDTRANSFORMER command:

ADDTRANSFORMER TRANSFORMERNAME SCRIPT

For example, to extend KAFY with GPT-2, we use the command:

ADDTRANSFORMER GPT-2 GPT2.PY

Once this command is invoked, a new transformer with the name GPT-2 will be added to the extensible transformer module in KAFY with its python implmentation script GPT2.PY that can be obtained from the HuggingFace library [6]. It is important to note that transformers are plugged-in as vanilla architectures, not as pretrained models. Once a new transformer is plugged-in, it needs to be trained for each operation it needs to support.

4 EXTENSIBILITY IN OPERATIONS

Since the main idea of KAFY is to replace training the transformer architecture with language statements by training it with trajectories, the output of the training process is a trajectory model not a language model. With this, KAFY leverages trajectory models as language models, and trains them for various trajectory operations, including summarization, generation, prediction, and classification. Then, the basic KAFY algorithm for a certain operation is just about calling the underlying trajectory model for that operation to get its answer. Unfortunately, this alone would be inaccurate for two reasons: (1) Transformers are not spatially aware and may produce infeasible outputs, and (2) Each trajectory operation may have its own special logic in terms of utilizing the underlying trajectory model. Both issues are addressed when extending KAFY with new operations via the following command:

ADDOPERATION OPERATIONNAME [SCRIPT] [CONSTRAINTS]
TRANSFORMERNAME([CONFIG]) [TRAININGDATA]

The OPERATIONNAME argument specifies the name of the trajectory operation that will be registered in KAFY . The SCRIPT argument is optional. If not defined, the new operation will use the basic KAFY algorithm, otherwise, it refers to a Python code file. The optional CONSTRAINTS argument provides a list of spatial constraints to ensure the transformer output is spatially feasible before being processed by the script. The TRANSFORMERNAME argument specifies which transformer the operation should use; it must be one of the transformers already registered in KAFY . The CONFIG argument is optional to allow specifying transformer parameters, e.g., such as number of layers and model dimensionality. The optional TRAININGDATA argument points to a CSV file for training the transformer. If not provided, the operation will still be registered, but will be unusable until its transformer is trained. For example, to extend KAFY with trajectory summarization, we use the command line:

ADDOPERATION SUMMARIZATION SUMMARIZATION.PY
T5 CONSTRAINTS.PY DATASET(TRAJECTORY,SUMMARY)

This indicates adding a new operation titled *Summarization* that employs the encoder-decoder transformer T5 with its default parameters after training it with a dataset that includes a list of pairs

of (Trajectory,Summary). The output of the model should be constrained per the rules defined in Constraints.py, which basically says that for an original trajectory T, the set of output summary points should be all part of T, preserve their order in T, include the last point in T, and have no duplicates. Summarization.py overrides the basic Kafy algorithm. Its main idea is that a trajectory summary must preserve the shape of the original trajectory by ensuring that a set of *anchor* points (i.e., end points and points with sharp turns in direction) must appear in the summary. Hence, the input script finds such anchor points, splits the T at each anchor point into a set of sub trajectories, calls the basic Kafy algorithm to summarize each sub trajectory, and finally concatenate the summarized sub trajectories to produce the final summary trajectory.

Once we register the new operation, we can train its transformer by any additional dataset that may cover different geographical areas through the following command:

TrainModel OperationName TrainingData

The OperationName argument specifies the name of the trajectory operation that we will train with the new dataset specified in the TrainingData argument. The operation must be one of the existing operations that were added earlier to Kafy through the AddOperation command. The training will be done for the transformer that was registered with this operation when it was added to Kafy . Finally, to run a registered operation in Kafy , we use the following command:

RunOperation OperationName([Args]) InputData.csv

The OperationName argument specifies the name of the trajectory operation that we need to run. Args is a set of optional parameters that will be passed to the procedure of the trajectory operation, which was defined in the Script argument of the AddOperation command. The InputData.csv argument is the input dataset for the trajectory operation, which would have a different format for each operation. To ensure accurate results, the input data should be geographically located in an area where the specified operation was already trained. If input data comes from an area where there is no prior training, Kafy will raise a lack of training data error and request to train a model in this area first. Once the RunOperation command is invoked, Kafy will do the following: (1) Retrieve the model that corresponds to the same area of the input data from the model repository, (2) execute the script along with the constraints that were provided to this operation when it was added to Kafy .

The first release of Kafy comes with five trajectory operations, namely, summarization, prediction, generation, imputation, and classification and three transformers, namely, BERT, GPT, and T5.

5 DEMO SCENARIOS

This section shows three demo scenarios where conference attendees can: (a) understand Kafy internals by extending it with new transformers and/or trajectory operations, (b) upload new training data to Kafy and see how it impacts the training models in various areas worldwide, and (c) interactively query Kafy by issuing various trajectory analysis tasks, and check their performance in terms of accuracy and execution time. The demo will be available for conference attendees through a web-based GUI powered by a real deployment of Kafy that is equipped with three transformers, five trajectory operations, and various real trajectory datasets.

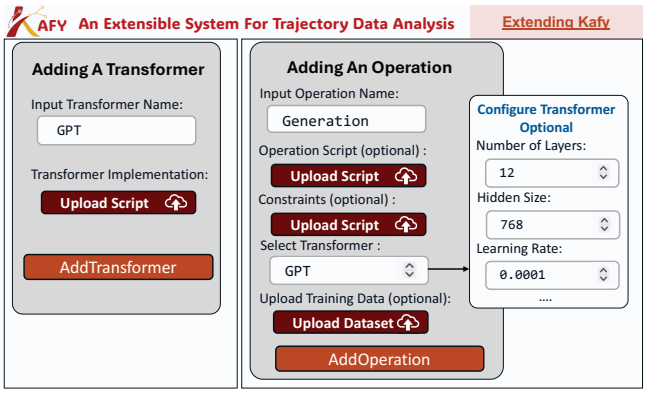

Figure 2: Demo Scenario 1: Extending Kafy

5.1 Demo Scenario 1: Extending Kafy

Figure 2 illustrates the demo scenario that allows conference attendees to extend Kafy with new transformer architectures and trajectory operations as explained below:

Adding a Transformer. Using the left panel in Figure 2, demo attendees can specify a TransformerName, upload a transformer implementation Python Script, and click on AddTransformer button. This will trigger the AddTransformer command line as described in Section 3, which registers the new transformer in Kafy and makes it accessible to all its trajectory operations. Conference participants will be provided with example transformers and their implementations, and are also encouraged to add their own transformers. This version of Kafy is already equipped with three transformers, namely BERT, GPT-2, and T5.

Adding an Operation. Using the right panel in Figure 2, demo attendees can specify an OperationName, (optionally) upload an operation Script, (optionally) define a set of spatial Constraints, select one of the transformers registered in Kafy , (optionally) update the default configuration parameters for the selected transformer, (optionally) upload a training dataset in a CSV file, and click on AddTransformer button. This will trigger the AddOperation command line as described in Section 4, which registers a trajectory operation with Kafy . Conference participants will be provided with example of trajectory operations, and are encouraged to add their own. This version of Kafy is already equipped with five trajectory operations, namely summarization, generation, prediction, imputation, and classification.

5.2 Demo Scenario 2: Uploading New Data

Figure 3 illustrates the demo scenario that allow users to upload new training data for any trajectory operation anywhere in the world, which will trigger the TrainModel command line as described in Section 4. Using the left panel, attendees can pick the operation(s) that they want to train with the new data, then upload a CSV file that includes the training data. One dataset may be used to train more than one operation. The format of the file is dependent on the selected operation. For example, in trajectory summarization, the CSV file includes pairs of trajectories and their summaries. For trajectory classification, the CSV file includes pairs of trajectories and their labels. For trajectory generation, prediction, and imputation, the CSV file just includes a set of trajectories. Hence, such CSV file

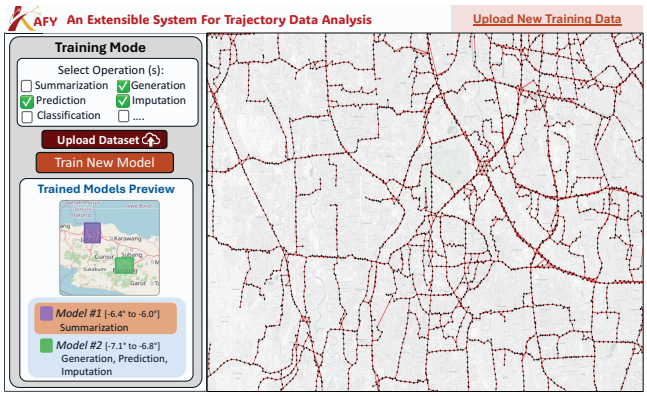

Figure 3: Demo Scenario 2: Uploading Training Data to KAFY

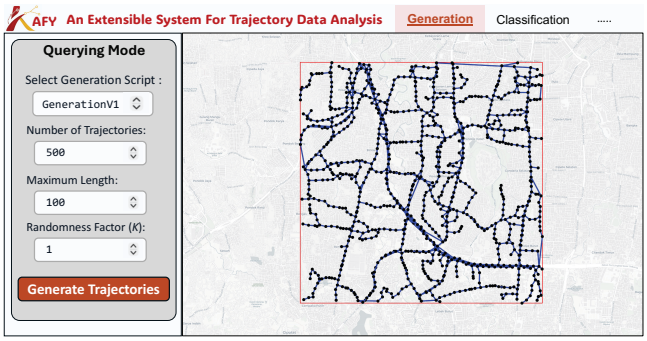

Figure 4: Demo Scenario 3.A: Trajectory Generation

can be used to train these three operations once, if the user marks them all. The map on right panel will show the uploaded data to visually ensure the upload accuracy.

Users can also have a preview of the current trained areas and models built in KAFY . The lower part of the left panel shows the available models on the map. In Figure 3, this shows two models depicted by purple and green boxes. Clicking on the purple box shows that it is a model for Jakarta, Indonesia, trained to support trajectory summarization using T5 transformer. The green box is for a model built for Bandung, Indonesia, trained to support trajectory generation, prediction, and imputation using GPT transformer.

5.3 Demo Scenario 3: Trajectory Analysis Tasks

This demo scenario will let conference attendees perform various trajectory analysis tasks using KAFY . This will be done through various GUI screens that will basically take input parameters, trigger the RUNOPERATION command line as described in Section 4, and visually show the result on the map.

Trajectory Generation. Figure 4 depicts the user interface for trajectory generation. On the left panel, attendees can select one of the trajectory generation operations in KAFY . Recall that users may extend KAFY with several versions of the trajectory operations using different transformers and/or scripts. Hence, the user needs to pick one of them. The user will also need to set the parameters for trajectory generation, which include the number of generated trajectories, the maximum length of each trajectory, and the area on the map where the user needs to generate trajectories in. The set of generated trajectories are then displayed in the map in an

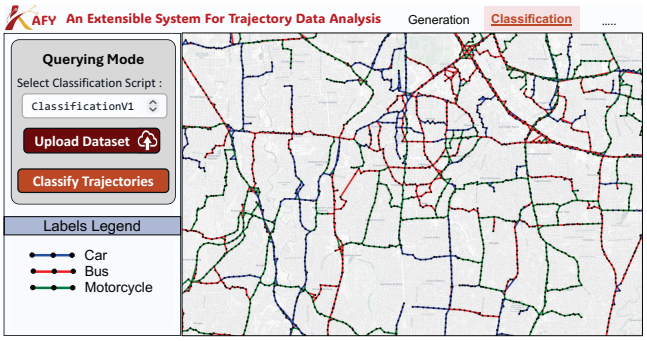

Figure 5: Demo Scenario 3.B: Trajectory Classification

interactive way. Users can change parameters and see the generated trajectories interactively.

Trajectory Classification. Figure 5 depicts the user interface for trajectory classification. On the left panel, attendees can select one of the trajectory classification operations in KAFY and upload the set of trajectories that need to be classified. The figure shows such trajectories on the map colored based on their class of either car, bus, or motorcycle. Also, a classification score is displayed on top of each trajectory to reflect how confident KAFY is in its labeling.

Trajectory Summarization, Prediction, and Imputation. Due to space limitations, we are not showing the screenshots of these operations. Yet, they are all pretty similar to the screenshots of Figures 4 and 5. In all cases, the user would need to pick the specific operation to execute and upload the trajectory data that need to be processed. In case of prediction, the user would need to set one more parameter, which is the number of points to predict. The result of these operations are interactively shown on the map. For summarization, we show the summarized trajectories on top of the original trajectories to see how accurate is the summarization. For prediction and imputation, we show the new trajectories after predication and imputation, yet, the predicted and imputed points are shown in different colors.

REFERENCES

[1] W. G. Aref and H. Samet. Efficient Processing of Window Queries in The Pyramid Data Structure. In *PODS*, 1990.
[2] I. Brodsky. H3: Uber's Hexagonal Hierarchical Spatial Index, 2018. https://eng.uber.com/h3/.
[3] P. Cudré-Mauroux and E. W. andvSamuel Madden. TrajStore: An Adaptive Storage System for Very Large Trajectory Data Sets. In *IEEE ICDE*, 2010.
[4] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv*, 2018.
[5] X. Ding, L. Chen, Y. Gao, C. S. Jensen, and H. Bao. UlTraMan: A Unified Platform for Big Trajectory Data Management and Analytics. *PVLDB*, 11(7), 2018.
[6] Hugging Face Library: GPT-2. https://huggingface.co/docs/transformers/en/model_doc/gpt2.
[7] H. Lan, J. Xie, Z. Bao, F. Li, W. Tian, F. Wang, S. Wang, and A. Zhang. VRE: A Versatile, Robust, and Economical Trajectory Data System. *PVLDB*, 15(12), 2022.
[8] A. Radford and K. Narasimhan. Improving language understanding by generative pre-training, 2018.
[9] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020.
[10] Z. Shang, G. Li, and Z. Bao. DITA: Distributed In-Memory Trajectory Analytics. In *ACM SIGMOD*, 2018.
[11] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler. Efficient Transformers: A Survey. *ACM Computing Surveys*, 55(6), 2023.
[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is All you Need. In *NeurIPS*, 2017.
[13] H. Wang, K. Zheng, X. Zhou, and S. Sadiq. SharkDB: An In-Memory Storage System for Massive Trajectory Data. In *ACM SIGMOD*, 2015.