ARGO: A Big Data Framework for Online Trajectory Prediction

Petros Petrou[1], Panagiotis Nikitopoulos[1], Panagiotis Tampakis[1], Apostolos Glenis[1], Nikolaos Koutroumanis[1], Georgios M. Santipantakis[1], Kostas Patroumpas[2], Akrivi Vlachou[1], Harris Georgiou[1], Eva Chondrodima[1], Christos Doulkeridis[1], Nikos Pelekis[1], Gennady L. Andrienko[3], Fabian Patterson[3], Georg Fuchs[3], Yannis Theodoridis[1], George A. Vouros[1]

[1]Data Science Lab, University of Piraeus (UPRC), Greece
[2]Information Management Systems Institute, Athena Research Center, Greece
[3]Fraunhofer Institute IAIS, Sankt Augustin, Germany
{ppetrou,nikp,ptampak,aglenis,koutroumanis,gsant,kpatro,avlachou,hgeorgiou,evachondrodima,cdoulk,npelekis, ytheod,georgev}@unipi.gr, {gennady.andrienko, Fabian.Patterson,georg.fuchs}@iais.fraunhofer.de

ABSTRACT

We present a big data framework for the prediction of streaming trajectory data, enriched from other data sources and exploiting mined patterns of trajectories, allowing accurate long-term predictions with low latency. To meet this goal, we follow a multi-step methodology. First, we efficiently compress surveillance data in an online fashion, by constructing trajectory synopses that are spatio-temporally linked with streaming and archival data from a variety of diverse and heterogeneous data sources. The enriched stream of trajectory synopses is stored in a distributed RDF store, supporting data exploration via SPARQL queries. The enriched stream of synopses along with the raw data is consumed by trajectory prediction algorithms that exploit mined patterns from the RDF store, namely medoids of (sub-) trajectory clusters, which prolong the horizon of useful predictions. The framework is extended with offline and online interactive visual analytics tool to facilitate real world analysis in the maritime and the aviation domains.

CCS CONCEPTS

Information systems → Information systems applications → Spatial-temporal systems → Data streaming

KEYWORDS

geostreaming, mobility events, trajectories, location prediction

ACM Reference format:

P. Petrou[1], P. Nikitopoulos[1], P. Tampakis[1], A. Glenis[1], N. Koutroumanis[1], G. M. Santipantakis[1], K. Patroumpas[2], A. Vlachou[1], H. Georgiou[1], E. Chondrodima[1], C. Doulkeridis[1], N. Pelekis[1], G. L. Andrienko[3], F. Patterson[3], G. Fuchs[3], Y. Theodoridis[1], G. A. Vouros[1]. 2019. ARGO: A Big Data Framework for Online Trajectory Prediction. In *Proc. of SSTD. ACM, Vienna, Austria, x pages.* https://doi.org/10.1145/1234567890

1 Introduction

As the maritime and aviation domains have major impact to the global economy, we need to advance the capacities of systems to improve safety and effectiveness of critical operations involving big fleets of moving entities in large geographical areas [10]. Towards this goal, the correlated exploitation of heterogeneous data sources offering vast quantities of archival and high-rate streaming data is important for increasing the accuracy of computations when predicting future states of moving entities. However, operational systems in these domains for predicting trajectories are still limited to a short-term horizon to date, while facing increased uncertainty and lack of accuracy in mobility data.

Motivated by these challenges, we present ARGO[1], a big data framework for the prediction of streaming trajectory data, enriched with data from other sources and exploiting mined patterns of trajectories. ARGO offers online predictions such as '*the estimated flights of all aircrafts around the globe over the next 10 minutes*' or '*the predicted routes of vessels in the next hour*' based on their current movement and historical motion patterns in the area. ARGO incorporates several innovative modules, operating in streaming mode over surveillance data, to deliver accurate long-term predictions with low latency requirements. Incoming streams of moving objects' positions are cleansed, compressed, integrated and linked with archival and contextual data by means of link discovery methods. All data is transformed to RDF, and eventually stored for offline processing and analysis in a custom-built, distributed RDF store, specialized to process spatio-temporal RDF data. While this online process feeds the trajectory prediction module with enriched trajectory synopses, the RDF store supports batch processing over vast-sized integrated data, and feeds modules performing trajectory clustering and visual analytics, which are used to discover mobility patterns, exploited for online trajectory predictions.

This paper describes the design and implementation of ARGO on top of state-of-the-art big data technologies (Spark, Flink, and Kafka), as well as comprehensive demonstration scenarios that clearly show its value on real-life data sets from the maritime and aviation domains. To our knowledge, in contrast to related state-of-the-art systems [11][7] and research approaches [4][8], ARGO is unique as a big data framework capable to provide long-term trajectory predictions in an online fashion.

2 Major Modules and System Architecture

In this section, we present the details of the major modules of ARGO. Then, we provide the overall architecture of our system.

SSTD'19, August, 2019, Vienna, Austria
© 2019 Copyright held by the owner/author(s). 978-1-4503-0000-0/18/06...$15.00
https://doi.org/10.1145/1234567890

[1] In Greek mythology, ARGO was the ship on which Jason and the Argonauts sailed to Colchis to retrieve the Golden Fleece. According to the legend, ARGO was at least as fast as a dove, thus metaphorically it was a 'flying' ship.

2.1 Synopses Generator

The Synopses Generator (SG) [2][3] provides online, summarized representations of trajectories. Usually, large amounts of raw positional updates of moving objects can hardly contribute additional knowledge about their actual motion patterns, e.g., most vessels normally follow almost straight, predictable routes at open sea, unless "emergency" circumstances prevail, such as adverse weather conditions, accidents, etc. Thus, instead of resorting to a costly trajectory simplification method, which retains every incoming position for each object, and since our ultimate goal is trajectory prediction, SG drops any easily predictable positions along trajectory segments of "normal" motion characteristics and reconstructs the traces of the moving objects approximately from the judiciously chosen *critical points* without harming the quality of the resulting approximation. More specifically, SG detects trajectory features from streaming positions, identifies significant changes in movement online, and outputs lightweight synopses of coherent trajectory segments. Critical points can be of various types, including start and end of gap/stop/slow motion/change in heading. Note that the critical points are emitted at operational latency, so as not to cause delays in subsequent processing. Hence, this derived stream of trajectory synopses keeps in pace with the incoming raw streaming data so as to get incrementally annotated with semantically important mobility events. In contrast to existing techniques, SG produces trajectory synopses in real-time, that are both space-efficient and highly accurate, with extremely low latency (within milliseconds since the arrival of raw messages [2][3]).

2.2 Semantic Integrator

The compressed stream of trajectory positions is received by the Semantic Integrator (SI), which performs two tasks: (a) data transformation to RDF, and (b) spatio-temporal link discovery (LD) against other data sources. The output of the SI is a stream of integrated data, representing enriched trajectory synopses [5].

Data transformation to RDF is performed using RDF-Gen, an efficient method that operates in a record-by-record fashion and outperforms state-of-the-art tools [12]. RDF-Gen exploits the concept of graph templates, which are populated at runtime using data values from a specific data source, in compliance with a given ontology. It supports various data sources (CSV, XML, ShapeFile, SPARQL endpoint, etc.), user-defined functions for data values transformation and operates in an online fashion. Its parallel version has been implemented in Apache Flink.

After the stream of positions has been transformed to RDF, the link discovery module is responsible for interlinking with other archival sources. The most challenging link discovery tasks are of spatio-temporal nature; discovering topological (e.g., within) and proximity relations (e.g., nearby) between a moving entity and various 2D and 3D spatial data sets (ports, fishing and marine protected areas, waypoints, airblocks, sectors, etc.). Also, relations between more complex types are supported (i.e. polyline, polygon). For efficiency, blocking techniques are adopted, practically organizing the target data set in a grid structure, thus assigning any incoming entity of the source data set to a single cell (topological relations) or few cells (proximity relations), thus drastically reducing the number of necessary comparisons. Compared to existing LD frameworks, the following innovative features are provided: (a) support for proximity relations, instead of only topological, (b) operating over streaming inputs, (c) more efficient filtering [13] that exploits grid cells with empty space, and (d) a scalable, data-parallel implementation on Apache Flink.

2.3 Data Manager

The Data Manager has as the fundamental module the distributed spatiotemporal RDF engine. It comprises two distinct layers: (a) the distributed storage layer, and (b) the parallel processing layer.

RDF storage engines typically store RDF data encoded using unique integer identifiers, for efficient indexing and access. In our work, we exploit a deliberate encoding scheme that maps spatiotemporal positions to 1D values (keys) [9], essentially providing lightweight indexing. In turn, this enables spatio-temporal filtering by checking against this key, avoiding the construction and maintenance overhead of distributed spatio-temporal index structures. After encoding the data, any NoSQL storage solution can be exploited. In our prototype, we store data in HDFS using Parquet, allowing compression and efficient retrieval. Also, we support different RDF storage layouts; both the *one-triples-table* approach, as well as *property tables* together with a *leftover triples* table to reduce the number of required joins at retrieval time. Moreover, the corresponding dictionary that maintains the mapping from 1D keys to string values is stored in Redis, as distributed, in-memory key-value pairs.

The processing layer, implemented in Apache Spark, takes as input SPARQL queries along with spatio-temporal constraints. We have implemented the main components of a query processing engine [1]: logical and physical operators, logical planner, physical planner, as well as main optimization techniques, such as join selection (repartition join vs. broadcast join) by output size estimation using histograms. This way, our engine supports different execution plans, and can be further extended towards cost-based optimization. Furthermore, we support predicate pushdown by exploiting the interplay between Spark and Parquet, thus avoiding reading all triples from disk to memory. Compared to existing distributed RDF engines that need a post-processing step to exclude triples that do not satisfy the spatio-temporal constraints, thus producing many candidate results in vain, we filter data "jointly", thus pruning more candidate triples at early an processing stage of an execution plan triples at an early processing stage of the execution.

2.4 (Sub-)Trajectory Clustering Module

The objective of the (Sub-)Trajectory Clustering (STC) module is to first partition trajectories into sub-trajectories and then identify the most representative ones that will act as cluster "pivots". In turn, by using these representatives, it forms clusters around them, while at the same time identifies those (sub-)trajectories (called outliers) that fit into no group. The S^{2}T-Clustering [4] solution to this problem first applies a neighborhood-aware trajectory segmentation method, where each trajectory is split into subtrajectories whenever the presence of other objects in its neighbourhood changes significantly; then a specialized sampling method is employed that selects the most representative (sub)trajectories to serve as the seeds of the clusters; and finally a greedy clustering algorithm is used that decides which of representatives can serve as the pivots of the clusters w.r.t. an optimization criterion. However, the work presented in [4] is centralized and does not scale for voluminous data sets. For this reason, we adapted and re-implemented S^{2}T-Clustering by using the MapReduce programming framework. The role of this module to the overall architecture of ARGO is to take as input data selected from the RDF store, apply STC, and provide the resulting representatives (i.e. cluster medoids) both to the prediction module as well as back to the RDF store. Figure 1 illustrates the visual exploration of cluster representatives using the visual analytics tool (V-Analytics) that is coupled with ARGO.

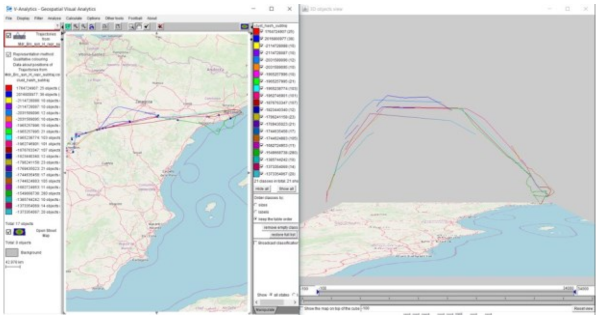

Figure 1: Visual exploration of clustering results in VAnalytics (offline) tool: 2D and 3D shapes of cluster medoids.

2.5 Future Location Prediction Module

The Future Location Prediction (FLP) module aims to make an accurate estimation of the next movement of a moving object within a specific look-ahead time frame, as shown in Figure 2. FLP is usually addressed at short-term horizon, since the prediction errors are cumulative and expand exponentially as the look-ahead span increases. However, given properly trained models from historic data in conjunction to clusters of trajectories discovered, ARGO can support long-term prediction, as well, identifying cases that exceed regular mobility patterns. The FLP module addresses two main orthogonal aspects for big data applications: online/offline operation; and short-/long-term prediction. The first refers to the time frame available for producing the prediction, while the second refers to the time frame of the prediction itself, i.e., the look-ahead time. The online long-term prediction, named FLP-L, which exploits on discovered routes from historic data using the STC module is the most challenging, compared to the online short-term prediction or FLPS, which improves the RMF algorithm [6].

The original RMF formulation is very promising for shortterm FLP, but it comes with several stability issues (due to the application of SVD to noisy and/or ill-conditioned input data, which is common in real world). In this work we developed an improved, online variation of RMF, coined RMF*, in order to address these instabilities, as well as real world requirements for the maritime and aviation domains in the context of big data. Our approach differs from the standard RMF framework in key aspects regarding sampling rate, type of processing (atomic vs. distributed), nature of processing (batch vs. online) and nature of data. We further extended RMF* with motion pattern discovery capabilities, hereafter called P-RMF*, selecting among candidate domain-specific motion functions the most promising in terms of prediction error to pre-calculate a solution for the motion state vector, avoiding SVD instability issues.

FLP-L exploits the cluster representatives mined from historical data as a reference for producing FLP forecasts "aligned" with the "closest-matched" route in the maximumlikelihood sense. This is a very different approach for tackling the FLP problem, as it makes the associated predictive models less adaptive but more reliable, by introducing specific "memory", based on historic data of either a large fleet of objects or a single object with long duration. In practice, this means exploiting the patterns and trends of objects in the same routes, conditions and constraints (e.g., regular maritime traffic). These routes are efficiently exploited to enhance the accuracy and size of the lookahead window. In should be noted that having a pre-computed set of routes available for retrieval and top-k lookup, although more computationally- and resource-intensive than the FLP-S approach described earlier, it is still fast enough to implement it in an online fashion. In practice, several such routes are pre-computed offline by first selecting the area of interest for several thousands of objects, making FLP-L feasible to provide long-term predictions for large fleets in an online fashion.

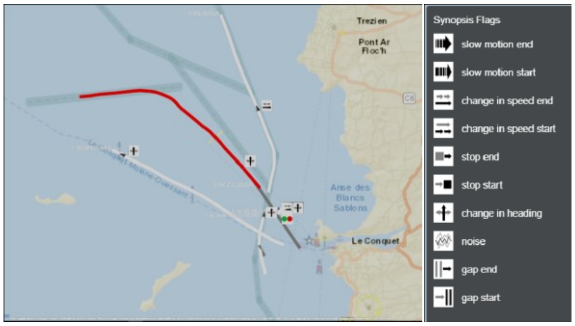

Figure 2: Interactive visual exploration of raw (red) vs. predicted (grey) location data in IVA (online) tool, along generated synopses (icons).

2.6 System Architecture

The ARGO framework for trajectory prediction is implemented as a big data architecture and illustrated in Figure 3. It comprises two parts/layers: (a) stream processing layer, and (b) batch processing layer, which interact in order to provide the desired functionality.

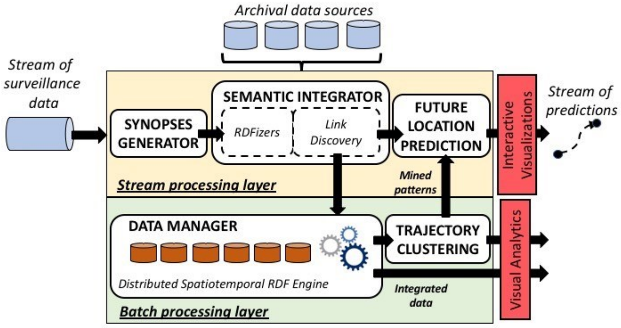

Figure 3: Architecture of the ARGO framework.

In brief, the stream processing layer processes the stream of surveillance data, and performs data cleaning, noise elimination, compression and semantic data integration, in an online manner. The synopsized and enriched data stream, represented in RDF, can be consumed as it is, thus enabling the deployment of data analysis pipelines, and it is also stored in a distributed spatiotemporal RDF store. This store supports scalable and efficient batch processing of SPARQL queries, providing filtered, integrated, spatiotemporal data for higher level analysis tasks. Offline analysis of integrated data (e.g., for trajectory clustering) generates mined patterns, which are exploited in conjunction to the enriched data stream during the online operation of the trajectory prediction module.

Stream processing layer: The stream-based interconnection is based on Apache Kafka to ensure scalability and faulttolerance. The individual modules have been developed in Apache Flink, thus allowing parallelization in a computer cluster. While the synopses generation and data transformation to RDF are easily parallelized by partitioning by moving object identifier, the parallelization of the link discovery module is more challenging. Two different techniques have been implemented: (a) blocking of archival data by space partitioning, and (b) building an index over archival data and broadcast this index to nodes.

Batch processing layer: The massive generation of integrated data in RDF poses requirements for scalable SPARQL querying. Existing parallel/distributed RDF stores cannot natively support spatiotemporal data. Motivated by this shortcoming, we developed a distributed spatiotemporal RDF engine in Apache Spark, which includes salient features, including 1D spatiotemporal encoding [9], logical/physical planning and optimization, thus supporting efficient processing of spatiotemporal SPARQL queries. This engine feeds the STC module with integrated data, filtered in space and time. The STC discovers movement patterns in the form of medoids of (sub-) trajectory clusters. Finally, the FLP exploits the mined patterns to predict the future positions of a moving object.

3 Demo Specifications

ARGO will be demonstrated using two real-world data sets. The first data set (maritime domain) covers a time span of six months, from October 2015 to March 2016 and provides positions of vessels sailing in the Celtic sea, and is publicly available[2]. The second data set (aviation domain) consists of radar tracks of the Spanish airspace for one week in April 2016. Our demonstration consists of three complementary parts: (1) preparation of surveillance data (cleaning, compression, integration) to be stored in the distributed RDF store, (2) interactive pattern discovery, which is based on the interplay between the visual analytics and trajectory clustering components on the one hand, and the distributed RDF store that provides query results for interactively selected spatio-temporal slices of integrated data, and (3) online trajectory prediction, exploiting the mined patterns.

Demonstration I: Preparatory phase. Initially, the user has the opportunity to comprehend the internals of our implementation and API, which bases on state-of-the-art big data frameworks, such as Apache Flink and Kafka. In detail, the user selects a data set and initiates the chain of stream processing components, which generates trajectory synopses, transforms data to RDF, and performs interlinking with archival data describing areas of interest (e.g., Natura protected areas and fishing areas for maritime, and 3D sectors and airblocks for aviation).

Demonstration II: Interactive pattern discovery. Having gained the necessary background knowledge, the user experiences an interactive session, where the visual analytics component selects slices of integrated data by querying the RDF store, filters and aggregates it, and repeats this operation until a suitable data set for training is found. Then, the trajectory clustering component processes this data set and outputs the discovered patterns in the form of representative trajectories.

Demonstration III: Online trajectory prediction. In turn, we present how the prediction module exploits the mined patterns to output predictions for the future location of incoming moving object positions from the stream of integrated data. Predictions get updated as new positional data arrive in the system.

For deeper comprehension of the demonstration scenarios, related videos are available at ARGO's demo web page[3].

ACKNOWLEDGMENTS

This work was partially supported by projects datACRON (grant agreement N. 687591), Track&Know (grant agreement N. 780754) and MASTER (Marie Sklowdoska-Curie agreement N. 777695) that have received funding from the EU Horizon 2020 Pr.

REFERENCES

[1] P. Nikitopoulos, A. Vlachou, C. Doulkeridis, G.A. Vouros. 2018. DiStRDF: Distributed Spatio-temporal RDF Queries on Spark. EDBT/ICDT Workshops.

[2] K. Patroumpas, E. Alevizos, A. Artikis, M. Vodas, N. Pelekis, Y. Theodoridis, 2016. Online Event Recognition from Moving Vessel Trajectories. GeoInformatica, 21(2), 389-427.

[3] K. Patroumpas, N. Pelekis, Y. Theodoridis, 2018. On-the-fly Mobility Event Detection over Aircraft Trajectories. In Proceedings of SIGSPATIAL.

[4] N. Pelekis, P. Tampakis, M. Vodas, C. Panagiotakis, Y. Theodoridis, 2017. InDBMS Sampling-based Sub-trajectory Clustering, In Proceedings of EDBT.

[5] G.M. Santipantakis, A.Glenis, K.Patroumpas, A.Vlachou, C.Doulkeridis, G.A. Vouros, N. Pelekis, Y. Theodoridis. SPARTAN: Semantic Integration of Big Spatio-temporal Data from Streaming and Archival Sources. Future Generation Computer Systems, to appear.

[6] Y. Tao, C. Faloutsos, D. Papadias, B. Liu. 2004. Prediction and indexing of moving objects with unknown motion patterns. In Proceedings of SIGMOD.

[7] P. Tampakis, N. Pelekis, N. Andrienko, G. Andrienko, G. Fuchs, Y. Theodoridis. 2018. Time-aware Sub-Trajectory Clustering in Hermes@PostgreSQL (demo paper). In Proceedings of ICDE.

[8] R. Trasarti, R. Guidotti, A. Monreale, and F. Giannotti. 2017. MyWay: Location Prediction via mobility profiling. Information Systems, 64, 350-367.

[9] A. Vlachou, C. Doulkeridis, A. Glenis, G. Santipantakis, G.A. Vouros. 2019. Efficient Spatio-temporal RDF Query Processing in Large Dynamic Knowledge Bases. In Proceedings of SAC.

[10] G. A. Vouros, C. Doulkeridis, G. Santipantakis, A. Vlachou, N. Pelekis, H. Georgiou, Y. Theodoridis, K. Patroumpas, E. Alevizos, A. Artikis, G. Fuchs, M. Mock, G. Andrienko, N. Andrienko, C. Ray, C. Claramunt, E. Camossi, A.-L. Jousselme, D. Scarlatti, J. Manuel, 2018. Big Data Analytics for Time Critical Mobility Forecasting: Recent Progress and Research Challenges. In Proceedings of EDBT.

[11] F. Wu, T.K.H. Lei, Z. Li, and J. Han. 2014. MoveMine 2.0: Mining Object Relationships from Movement Data. In Proceedings of VLDB.

[12] G.M. Santipantakis, K.I. Kotis, G.A. Vouros, C.Doulkeridis. 2018. RDF-Gen: Generating RDF from Streaming and Archival Data. In Proceedings of WIMS.

[13] G.M. Santipantakis, A.Vlachou, C.Doulkeridis, A.Artikis, I.Kontopoulos, G.A. Vouros. 2018. A Stream Reasoning System for Maritime Monitoring. In Proceedings of TIME.

[2] http://doi.org/10.5281/zenodo.1167595

[3] URL: http://www.datastories.org/argo/demo.