SPARTAN: Semantic Integration of Big Spatio-temporal Data from Streaming and Archival Sources

Georgios M. Santipantakis, Apostolos Glenis, Kostas Patroumpas, Akrivi Vlachou, Christos Doulkeridis, George A. Vouros, Nikos Pelekis, Yannis Theodoridis

School of Information and Communication Technologies

University of Piraeus

Karaoli and Dimitriou 80, 18534 Piraeus, Greece

Abstract

An ever-increasing number of applications in critical domains, such as maritime and aviation, generate, collect, manage and process spatio-temporal data related to the mobility of entities. This wealth of data can be exploited for various purposes, towards improving the safety of operations, reducing economical costs, and increasing dependability: The major issue to achieve these objectives is increasing predictability of moving objects' trajectories and events. To achieve this purpose in a data-driven way we need to exploit in integrated manners data from a variety of disparate and heterogeneous data sources, both streaming and archival, regarding – among other – surveillance, weather, and contextual data. Motivated by this fact, in this paper, we propose a framework for semantic integration of big mobility data with other data sources that are necessary to data analytics tasks, providing a unified representation of such data. Notable features of our framework include the real-time generation of data synopses of moving entities' trajectories, the efficient and flexible transformation of data from heterogeneous and big data sources in RDF, and the spatio-temporal link discovery between spatio-temporal entities in diverse data sources. The design and implementation of our framework uses big data technologies (Apache Flink and Kafka), and our experimental evaluation demonstrates the efficiency and scalability of the proposed framework using large, real-life datasets.

Keywords: RDF generation, semantic stream generation, semantic integration, trajectory summarization, spatio-temporal link discovery, big spatio-temporal data

1. Introduction

The ever-increasing size of spatio-temporal data and the unprecedented rate of data generation from a wide variety of sources regarding the situation awareness and monitoring in critical domains raise the need for scalable, real-time management and analysis of mobility data. Several data analysis tasks rely on moving entities' *trajectories*, while trajectory detection and prediction are typically used to optimize everyday, real-life operations. However, using only the kinematic information provided by surveillance sources is far from sufficient, when at the same time a wealth of other sources, including for instance, weather and contextual[1] information is available too. Consequently, one of the major challenges is to enrich surveillance data, providing meaningful information about moving entities' trajectories, also annotating trajectories with related events, thereby creating *enriched trajectories* [1, 2]. Addressing this challenge calls for real-time processing and semantic integration of surveillance data with other, streaming and archival, data sources [3].

Our work is motivated by the need to advance the management and integrated exploitation of voluminous and heterogeneous data-at-rest (archival data) and data-in-motion (streaming data) sources, so as to significantly promote safety and effectiveness of critical operations for large numbers of moving entities in large geographical areas. Challenges throughout the Big Data ecosystem, with special focus on surveillance systems, concern effective detection and

[1]By contextual data, we refer to data other than surveillance and weather data, including regulated areas, moving object registries, archival spatial data (coastal maps, ports, fishing areas), routes, sector configurations.

prediction of moving entities' trajectories and forecasting of complex events associated to these trajectories. These challenges emerge as the number of moving entities and data sources increase at unprecedented scale. This results in generating vast data volumes, of heterogeneous nature, at extremely high rates, whose exploitation calls for novel big data integration techniques that will facilitate advanced data analytics.

In this paper, we propose SPARTAN[2], a big data framework that ingests streaming spatio-temporal data in real-time, extracts useful information, performs data cleaning and summarization, transforms data to RDF in compliance with a generic ontology for trajectories (also connected to domain aspects and domain-related data sources), and performs integration of surveillance data with other streaming and archival data sources. As a result, enriched surveillance data is produced, associating mobility data with other data, thereby offering opportunities for higher level analysis tasks, such as trajectory prediction and complex event recognition and forecasting to achieve higher levels of accuracy. In technical terms, we provide an efficient and scalable implementation of the proposed framework on top of parallel data processing platforms, based on Apache Flink and Kafka.

In more concrete terms, SPARTAN introduces the following innovative features in the integration process for mobility data, considering trajectories to be "first-class entities": (a) an online trajectory compression technique that produces accurate and compact trajectory synopses in real-time, in contrast to existing works that do not create synopses within milliseconds (or a few seconds at most) since the arrival of raw messages [4, 5], (b) an efficient data transformation method from heterogeneous sources to RDF, offering flexibility and consuming data from a wide variety of input sources, and (c) a spatio-temporal link discovery mechanism that integrates trajectory data with other contextual and weather data using spatio-temporal relations; an issue largely overlooked in the state-of-the-art frameworks for link discovery [6, 7] (see also [8] for a

[2]Semantic integration of big sPatio-temporal dAta fRom sTreAmiNg and archival sources.

recent survey). Moreover, all the above innovations are provided as an integrated prototype that consumes streaming (and archival) data and operates in real-time.

In summary, this paper makes the following contributions:

- We propose a big data framework for the provision of streaming mobility data, transformed in RDF and enriched with other data sources, with low latency requirements. Our framework entails the following specific innovations:
  - We show how to compress surveillance data in an online fashion, by constructing trajectory synopses that are both space-efficient and highly accurate, with low latency.
  - We present an efficient and flexible data transformation tool that accesses heterogeneous streaming and archival data from a variety of diverse data sources and generates RDF graph fragments in compliance with the datAcron ontology [9, 10].
  - We propose a generic spatio-temporal link discovery module that operates on streaming data, and efficiently discovers spatio-temporal relations, while supporting blocking techniques and different evaluation functions.
- We evaluate our approach experimentally using a prototype implementation on top of big data technologies and real-life data, thereby providing evidence about the efficiency of the framework and its potential to provide enriched RDF streams of surveillance data.

The rest of this paper is structured as follows: Section 2 reviews the related work and clarifies how our work advances the state-of-the-art. Section 3 presents the targeted problem setting and motivates our work. Section 4 crisply describes the datAcron ontology for the representation of semantic trajectories. Section 5 describes the overall semantic integration framework, and delves into the details of its components. Section 6 provides technical details on our prototype implementation using big data technologies. Section 7 demonstrates the efficiency of our framework by means of experimental evaluation using real-life datasets. Section 8 provides a discussion on how SPARTAN can be exploited for improving data analysis tasks. Finally, Section 9 concludes the paper and sketches future research directions.

2. Related Work

There are efforts on semantic integration of streaming with archival data designed to operate on RDF, such as [11], [12], or efforts towards a framework for the integration of distributed heterogeneous streaming and stored data sources through ontological models, e.g. in [13]. Recently, in [14, 15], an approach for integration of streaming with static relational data has been proposed. The Graph of Things [16] targets an IoT setting where many sources provide data for integration and querying, and supports spatial and temporal data, but it is not optimised for mobility data. To the best of our knowledge, none of the existing approaches targets streaming mobility data explicitly: We aim at providing enriched RDF streams of mobility data, revolving around the notion of trajectory, operating on streams of surveillance data, with low latency. In doing so, we do not use any stream query approach, e.g. [11, 17, 18], nor ontology-based access methods e.g. [13, 14], which may also make possible the integration of streaming with archival data sources, to the expense of introducing additional latency. Our aim is to provide enriched RDF streams of mobility data in online fashion and low latency, which can be consumed/exploited by analytics components following any of these, or other stream access approaches. Thus, our work goes one step further by introducing a unified framework for providing enriched streams of mobility data, incorporating online compression, data transformation, and link discovery functionality. The applicability of this framework is not limited to any combination of streaming or archival input sources.

2.1. Representation of Semantic Trajectories

Existing approaches for the representation of semantic trajectories either (a) use plain textual annotations instead of semantic links to other entities [19, 20, 21], hindering the provision of semantic information associated with moving objects' behaviour; (b) constrain the types of events that can be used for structuring a trajectory [19, 20]; or (c) make assumptions on the constituents of trajectories [21, 22, 23] (e.g. semantic trajectories in [21] are sequences of sub-trajectories, while in [23] are sequences of episodes).

To a greater extent than previous proposals, the datAcron ontology used here *supports the representation of enriched trajectories at multiple, interlinked levels of analysis*: For instance, although [23] provides a rich set of constructs for the representation of semantic trajectories, these are restricted to be sequences of episodes, each associated with raw trajectory data, and optionally, with a spatio-temporal model of movement. However, there is no fine association between abstract models of movements and raw data. On the other hand, [21] provides a two-levels analysis where semantic trajectories are lists of semantic sub-trajectories, and each sub-trajectory in its own turn is a list of semantic points. Regarding events and episodes, these are connected to specific resources at specific levels of analysis: In [21] events -mostly related to the environment rather than to the trajectory itself- are connected to points only (something that may lead to ambiguities in some cases), while in [23] episodes concern things happening in the trajectory itself, and may be associated to specific models of movement: It is not clear how multiple models of a single trajectory -each at a different level of analysis- connected to a single episode, are associated. Finally, contextual information in [23] is related to movement models, episodes or semantic trajectories, which is quite generic, while in [21] environment attributes are associated to points only, and are assigned specific values.

The datAcron ontology[3] has been presented in [9, 10], where we have shown that it supports data transformations that are required by analytics tasks, pro-

[3]`http://ai-group.ds.unipi.gr/datacron_ontology`

viding information of the appropriate form at various levels of analysis.

2.2. Trajectory Summarization

Since trajectory synopses detect changes in mobility features from positional streams, they essentially perform a kind of path simplification in online fashion. Hence, algorithms like the well-known Douglas-Peucker algorithm [24] up to more recent techniques like [25, 26] that operate in batch fashion cannot be considered, since they require knowledge of all points (i.e., entire trajectories) before applying any simplification.

In contrast, due to the high arrival rate of incoming streaming locations, trajectory summarization must be performed online. Ideally, samples should keep each compressed trajectory as much closer to its original course, chiefly by minimizing approximation error as in trajectory fitting methods [27, 28]. For instance, the sliding window approach in [28] keeps simplifying points along a line until the error exceeds a given threshold. In addition, they employ the speed difference between sub-trajectories as an error metric for retaining sample points. From another perspective, the STTrace algorithm [29] uses the concept of safe areas to generate a simplified trajectory using predefined speed and direction error bounds, arguing that a sample should be included in the approximate path as long as it reveals significant change in movement.

TrajStore [30] offers capabilities for indexing, clustering, and storing trajectory data. Based on an adaptive grid partitioning scheme, it is mostly geared towards queries retrieving data about many trajectories passing through a particular location. From another perspective and mainly focusing on savings in communication cost, dead-reckoning policies like [31] and mobility tracking protocols in [32] may be employed on board of the moving objects to relay positional updates only upon significant deviation from the course already known to a centralized server.

Recently, a bounded quadrant system was introduced in [33] in order to establish a rectangular bounding box as well as two bounding lines for each of its quadrants. Various compression error bounds can be estimated via convex hulls formed by the box and the bounding lines around all points. Further, the one-pass, error-bounded algorithm suggested in [34] is based on a novel local distance checking method and involves several optimizations in order to achieve higher compression. It also introduced a more aggressive variant, which allows "patches" by interpolating new points when objects have sudden changes in their paths or relay updates intermittently.

To the best of our knowledge, none of the aforementioned techniques has ever been applied specifically on trajectories of vessels or aircrafts. In [5, 4] we have introduced a novel trajectory summarization framework specifically for online maritime surveillance. This framework consumes a geospatial stream of AIS tracking messages from vessels, it continuously detects important features that characterize their movement and subsequently recognizes complex events such as suspicious vessel activity. Making use of a sliding window over the incoming positions, this technique is able to detect either *instantaneous trajectory events* by simply checking potentially important sudden changes with respect to the previously reported location (e.g., a sharp change in heading) or *long-lasting trajectory events* are deduced after examining a sequence of instantaneous events over a longer (yet bounded) time period (e.g., a smooth turn). Examination of these long-lasting events is the one that yields the critical points in the resulting synopses. Once the window slides forward, expiring critical points are issued as results. So, results get reported periodically (upon window slides) with all recent "delta" changes, i.e., critical points evicted from the window. Extensively tested in maritime surveillance scenarios, this summarization methodolody was capable of handling scalable volumes of streaming vessel positions with up to 10,000 locations/sec. Not only has this algorithm shown that it could yield a compression ratio better than 95% and sometimes even 98% over the raw data, but most importantly, it also annotated the identified critical locations with movement characteristics. Further improving this approach, in this paper, we have performed a complete reengineering of the entire approach so as to enable scalable execution in cluster platforms. But in fact, we have significantly modified and enhanced the detection criteria for most mobility features, and also introduced new ones specifically for aircraft trajectories. Most importantly, we have now prescribed that critical points should be emitted at *operational latency*, i.e., within milliseconds (or a few seconds at most) since the arrival of raw messages. This invalidates all previous processing that employed time-based sliding windows with ranges up to several hours. Further, we can support multiple annotations per location in order to capture as much as possible all mobility-related information in a given critical point.

2.3. RDF Generation

There are many RDF generators that have been developed[4], most of them tailored to specific data formats (e.g. CSV, JSON, XML, XLS, etc), or even the vocabulary to be used [35]. Furthermore, solutions are usually tailored to a specific data format without considering the requirement for multiple source formats. Our approach [36] for generating RDF data aims to reduce the need to multiple RDF generators, providing a coherent and easy-to-be-tuned framework, further enabling its incorporation to widely used workflows, resulting to alleviating efficiency, maintenance and verification of results limitations of other approaches.

RDF Mapping Language (RML) [37] provides a mapping language for converting JSON, CSV, XML and HTML files to RDF, implemented on top of the open-source framework Sesame. RML has been proposed as an extension to R2RML [38], for the conversion of RDB to RDF. Although this solution can be extensible, e.g. custom functions can be defined for data conversion, functions that enable communication between RDF generators, are difficult to be defined. Also, verification of results and maintenance can be done only by RML experts. Contrary to that, we use standard and elementary RDF constructs (graph templates) which are considerably easier to learn, maintained and incorporated into RDF and SPARQL workflows.

Datalift [39], closer to our work, does not support incorporating custom

[4]ConverterToRdf: `https://www.w3.org/wiki/ConverterToRdf`, accessed 19/07/2017.

functions. Datalift can parse CSV, RDF, XML, RSS, GML and ESRI shapefiles (archival sources). The assumption that the vocabulary that best describes the data is a decision of data providers is a strong assumption, making also maintenance and verification of results a difficult task.

Finally, SPARQL-Generate [40], is the closest work to our proposed solution. SPARQL-Generate aims to introduce an extension to SPARQL 1.1 following the idea of SPARQL CONSTRUCT constructor, and generates triples according to a graph template. It provides constructors such as ITERATOR, useful for processing XML sources. The reliance on a SPARQL engine can introduce considerable latency issues, while only archival sources are considered. Although custom functions may be used, communication among SPARQL-Generate instances is not supported, resulting to potentially complicated workflows in cases where more than one sources are necessary to be processed concurrently.

2.4. Spatio-temporal Link Discovery

Even though the topic of link discovery has attracted much interest and attention lately (see [8] for a recent survey), there is not much work on the challenging topic of spatio-temporal link discovery.

2.4.1. Link Discovery Frameworks

LIMES [6] is a generic link discovery (LD) framework for metric spaces that uses the triangular inequality in order to avoid processing all possible pairs of objects. For this purpose, it employs the concept of exemplars, which are used to represent areas in the multidimensional space, and tries to prune entire areas (and the respective enclosed entities) from consideration during link discovery.

SILK [7] is a link discovery framework that proposes a novel blocking method called MultiBlock, which uses a multidimensional index in which similar objects are located near each other. In each dimension the entities are indexed by a different property or different similarity measure. Then, the indexes are combined together to form a multidimensional index, which is able to prune more entities by taking into account the combination of dimensions. It is the only generic LD framework to support the discovery of topological relations.

HR3 [41] and HYPPO [42] address link discovery tasks when the property values that are to be compared are expressed in an affine space with a Minkowski distance. Both approaches are designed to be efficient and lossless. In addition, HR3 [41] comes with theoretical guarantees on reduction ratio, a metric that corresponds to the percentage of the Cartesian product of two datasets that was not explored before reporting the link discovery results.

All these frameworks do not explicitly focus on spatio-temporal link discovery, nor do they address directly streaming data sources.

2.4.2. Link Discovery for Topological Relations

Most approaches for link discovery between two sets of regions A and B apply grid partitioning (a.k.a. space tiling) on the two sources, in order to perform efficiently the *filtering step*, avoiding the comparison of all regions in A to all regions in B. Then, in the *refinement step*, different optimizations aim at minimizing the number of computations necessary to produce the correct result set.

Smeros et al. [43] study link discovery on spatio-temporal RDF data. The authors study several topological relations that are defined on polygons. The topological relations do not take into account proximity nor distance between polygons, and several of those are meaningful only to polygons. The algorithm provided creates an equi-grid, and filters out cells that contain polygons that cannot satisfy the relation. The main drawback is that regions can be assigned to multiple cells, thus the same relations are discovered more than once and only at the final step duplicates are eliminated. This increases the computational cost of link discovery.

ORCHID [44] studies the problem of discovering all pairs of polygons between two sets A and B, such that their Hausdorff distance (practically Max-Min distance) is below a certain distance threshold. ORCHID employs a grid partitioning of the 2D space (space tiling). Based on this, ORCHID performs filtering and needs to compare only few regions to other regions. At the refinement step, ORCHID employs two techniques to reduce the number of distance computations necessary. First, it employs bounding circles as approximations of polygons (instead of Minimum bounding rectangles), and makes the observation that if the minimum distance between any pair of points in A and B respectively is larger than a threshold, then these regions cannot be linked. Second, it uses the triangular inequality and already computed distances to avoid computing new distances, thus pruning regions without distance computations.

RADON [45] is the most recent approach for discovering topological relations, and can discover efficiently *multiple* relations. RADON has the following main techniques: (a) it defines and computes the *Estimated Total Hypervolume* (ETH) of a set of geometries, which essentially quantifies the total size of the geometries present in a dataset, and chooses to index the dataset with smallest ETH value, (b) it uses space tiling and assigns geometries to cells, but applies an *optimized sparse space tiling*, which practically assigns all geometries of the first dataset (say A, with smaller ETH value) to cells, but only those geometries of the second dataset B that correspond to cells already occupied by geometries of A, and (c) it performs the filtering step using the Minimum Bounding Box (MBB) of the geometries, and avoids processing some of the topological relations (e.g., first checks if two regions are disjoint, and only evaluates other relations if they are not disjoint), while also employing a caching mechanism to avoid re-computing pairs of geometries (a, b) (where $a \in A$ an $b \in B$).

In summary, contrary to SPARTAN, most of the above papers target spatial rather than spatio-temporal data. Moreover, the type of relations supported is very restricted, mainly focusing on topological relations. Instead, in SPARTAN, proximity relations (e.g., *nearby*) are very important, and we design efficient algorithms for this case.

3. Motivation & Problem Setting

Trajectory-based operations, which involve spatio-temporal data of moving entities, have become increasingly important in real-life applications, as they lead to increased safety and minimize cost [46, 47]. Key issue to achieve these targets is increasing predictability of trajectories and if events related to the behaviour of moving entities. Thus, several analysis tasks revolve around trajectories, including future location and trajectory prediction as well as complex event recognition and forecasting. However, the existing operational systems mainly work at the level of trajectories, largely overlooking other data (weather, contextual, etc.) that can be combined with movement data, in order to further advance accuracy of predictions.

In the following, we focus on two domains of interest – maritime and aviation – and motivate the need for semantic integration of streaming with archival data sources, aiming at providing enriched data representations that facilitate higher level data analysis tasks.

3.1. Maritime Domain and Data Sources

Several maritime operations can benefit from real-time exploitation of enriched surveillance data, including identifying fishing zones, detecting illegal activity (e.g., smuggling, loitering, disobeying speed limitations), timely detection of dangerous situations leading to immediate response and rescue operations. All such operations rely on information about moving entities' trajectories. Trajectories can be detected by exploiting vessel position reports streamed from multiple sensors, terrestrial and satellite, as for example AIS[5] data. Nevertheless, the value of such kinematic information is substantially increased when enriched with other data.

Weather reports and forecasts are available from the National Oceanic and Atmospheric Administration (NOAA)[6], including global meteorological and oceano-

[5]Automatic Identification System

[6]`http://www.ndbc.noaa.gov/data`

graphic datasets from cooperating networks of ships and buoys. Other sources that provide information on aspects that affect or that are being affected by the mobility of entities – called contextual data sources – include: regulated fishing areas, FAO (Food and Agriculture Organization) major fishing areas[7] (the boundaries of which were determined in consultation with international fishery agencies on various considerations), areas with speed limitations (for example in a wider harbor area), fishing fleet register, the World Port Index (WPI), marine protected areas (Natura2000). This data is is available in various formats: binary GRIB files describe weather forecasts, while ESRI shapefiles are used to describe geographical areas of interest. Auxiliary CSV files are also used, relating FAO codes (and their subdivisions) with the corresponding names.

3.2. Aviation Domain and Data Sources

In the aviation domain, detecting and increasing the predictability of moving entities (aircraft) trajectories are major issues. This task is useful to be performed in an off-line setting, but also in real-time. To assist this task, real-time surveillance data coming from different networks (ADS-B, IFS, etc.) need to be combined with weather reports, as well as historical flight plans, trajectories produced by mathematical models, and aircraft databases. As in the maritime domain, real-time semantic integration produces enriched trajectories that can be used to improve the accuracy of various data analysis tasks (e.g., clustering) leading to more accurate predictions.

An area of interest where increased accuracy of trajectory predictions has a tremendous effect is Flow Management. The aim here is to ensure an optimum flow of air traffic to or through areas with respect to their capacity. On each operation day, the flow management monitoring process analyses periodically (typically every 20 minutes) the demand for each sector (comprising airspace volumes), by counting the expected number of flights in the sector during the next period (typically one hour). If a potential demand versus capacity imbal-

[7]`http://www.fao.org/fishery/area/search/en`

ance is detected (a hotspot), a regulation may be applied to adjust the demand values to the available capacity. Additional data sources that need to be taken into account include flight plans, historical trajectories, weather reports, issued regulations, and airspace information that describes the division of space to sectors. All this data is provided in different formats, including CSV files, AIXM[8] (XML-based), DDR flight plans (textual format), and binary GRIB files for weather forecasts.

3.3. Problem Setting

Given this wide variety of available data sources, a main objective of our work in the context of the datAcron project is to derive an *enriched stream* of mobility data, which can be consumed by other modules in an online fashion. Such consumer modules include analytics components, such as trajectory predictors and complex event recognition engines, and we refer to [48] for a detailed overview of the overall architecture of datAcron. The primary data in the derived stream is surveillance data of moving objects. This data is enriched with archival data sources, and transformed in a common representation using RDF. The derived stream of RDF can be consumed by online data analytics components, and can also be stored in a distributed RDF store for offline querying. However, these issues are outside the scope of this paper. Instead, in this paper, we present the techniques for online compression, data transformation to a common representation format, and linking with external data sources, thereby producing an enriched stream of mobility data.

4. The datAcron Ontology

The datAcron ontology[9] was developed to be used as a core ontology for the Maritime Situation Awareness (MSA) and Air Traffic Management (ATM) domains, towards supporting analysis tasks exploiting trajectories at various

[8] `http://www.aixm.aero`

[9] Documentation is available online at `http://ai-group.ds.unipi.gr/datacron_ontology`

levels of analysis. Its development has been driven by ontologies related to our objectives (e.g. DUL[10], SimpleFeature[11], NASA Sweet[12] and SSN[13]), as well as schemas and specifications regarding data sources from the different domains.

4.1. Definitions

Starting from the definitions of *raw*, *structured* and *semantic trajectories* provided in [1], a *raw trajectory* is a temporal sequence of raw data specifying the moving object's spatio-temporal positions. Raw data can be aggregated, analyzed and semantically annotated, providing multiple abstractions of a trajectory.

A *structured trajectory* (simply, trajectory) consists of a sequence of *trajectory parts* that can be either *raw positions* reported from sensing devises, aggregations of raw positions referred as *semantic nodes* or simply *nodes*, or *trajectory segments*.

A *semantic node* provides a meaningful abstraction or aggregation of raw positions, e.g. a set of raw positions may signify a "turn" event, represented as a single semantic node associated to the resource representing the "turn" event. A *trajectory segment* is a trajectory itself, part of a whole trajectory. Segmentation of trajectories can be done with different objectives depending on the application and target analysis. Any trajectory part may be associated with a co-occurring event. For example, a bad weather region may co-occur with a trajectory crossing-it (thus, related spatially) during a time period (related temporally).

A *semantic trajectory* is a meaningful sequence of trajectory parts, signifying events, activities, goals, etc. of moving entities.

[10]`http://www.ontologydesignpatterns.org/ont/dul/DUL.owl`
[11]`http://www.opengis.net/ont/sf`
[12]`https://sweet.jpl.nasa.gov/`
[13]`https://www.w3.org/2005/Incubator/ssn/ssnx/ssn`

4.2. Core Vocabulary and Overall Structure

According to the datAcron ontology, a trajectory (`Trajectory`) can be segmented to trajectory parts (`TrajectoryParts`), each including other segments and/or more semantic nodes, as illustrated in Figure 1. Each semantic node may be associated with a specific raw position or a temporally ordered sequence of raw positions of a moving object.

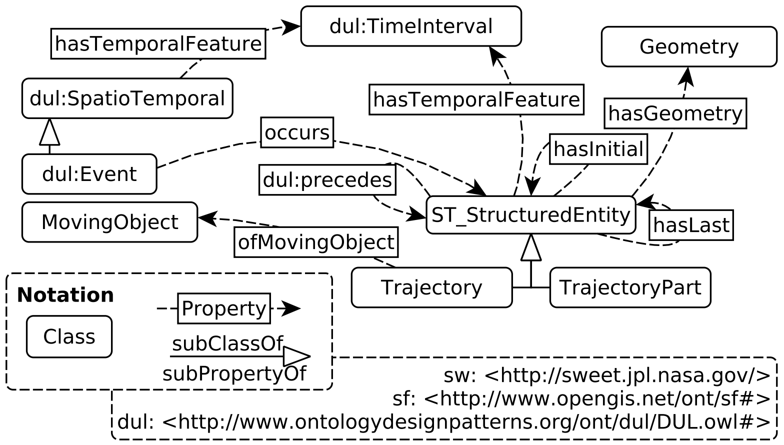


Figure 1: The main concepts and relations of the proposed ontology.

Trajectories and trajectory parts can be associated with contextual information, as well as with events (`dul:Event`). Although events may occur independently from the trajectory but co-occur with the trajectory, we focus on events on the trajectory itself (e.g. a "turn" or a "gap of communication") and to moving object's information (e.g. vessel in a protected or in a bad-weather area). Patterns for the specification of events and their associations to trajectory parts are presented subsequently.

4.3. Patterns of Semantic Trajectories

Figure 2 illustrates the generic pattern of raw and structured trajectories. The main concept in this pattern is the `Trajectory`, which is a subclass of Spatio-Temporal Structured Entity (`ST_StructuredEntity`). This, being a subclass of `dul:Region` represents a region in a dimensional space and time, used as a value for a quality of an Entity, while it also represents (structured) trajectories and their parts. A structured trajectory, as well as any of its parts, can be a temporal sequence of `TrajectoryPart` entities. As Figures 1 and 2 show, any trajectory and trajectory part, being an `ST_StructuredEntity`, can be associated to any other trajectory or trajectory part via a `dul:precedes` or a `dul:hasPart` property. This is further explained in the next paragraphs.

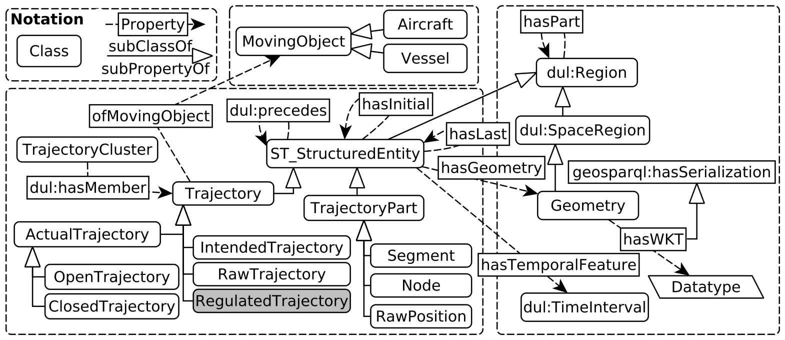


Figure 2: The pattern of structured trajectories. Domain specific concepts in gray

Direct subclasses of `Trajectory` are the:

- `IntendedTrajectory`: planned trajectories build by an `dul:InformationEntity` such as a `FlightPlan`,

- `ActualTrajectory`: trajectories constructed from actual positioning data, after some processing of the raw positional data,

An `ActualTrajectory` can be further distinguished to a `ClosedTrajectory` (i.e. a trajectory that has reached its destination) and to an `OpenTrajectory` (i.e. a trajectory in progress).

The `TrajectoryPart` can be further distinguished to one of the following subclasses:

- `Segment`: associated to a spatial region and a time proper interval.

- `Node`: associated to a point in space and a time instant or time period. The latter holds in case the node aggregates several raw positions. A `Node` can be the result of a data processing component computing compressions or aggregations of the raw positioning data.

- `RawPosition`: represents the raw (unprocessed) positioning data. Each raw position instance is associated to a point in space and a time instant.

A specific trajectory, as well as any of its trajectory parts, being instances of `dul:Region` can be associated to their parts via the `dul:hasPart` property or via the subproperties `hasInitial`, `hasLast` which indicate the first and last part of the `ST_StructuredEntity`, respectively. For instance, a trajectory may comprise a sequence of trajectory segments that (in turn) comprise other segments, nodes, or raw positions, and so on. The temporal sequence of structured entities is specified by means of the property `dul:precedes`. Trajectories related via the property `dul:precedes` represent subsequent trajectories of a specific object, and thus keep a long history of its movement.

Each structured entity (i.e. trajectory or trajectory part) can be associated to a specific *geometry* (`sf:Geometry`), representing a point or region of occurrence, and a *temporal entity* (`dul:TimeInterval`) specifying a time interval of occurrence. The Geometries of structured entities can be serialized into Well-Known-Text (WKT) and asserted as values to the property `hasWKT`, which is sub-property of `geosparql:hasSerialization`.

Finally, trajectories can be members of `TrajectoryCluster` entities, via the `dul:hasMember` property.

Towards the specification of semantic trajectories, trajectories are associated with events and contextual information. Specifically, each trajectory and trajectory part, being instances of `ST_StructuredEntity`, can be associated via the property `occurs` with events, as illustrated in Figure 3. An event can be associated with other events via the properties `dul:hasConstituent` or `dul:hasPart`: This is the case for high-level events associated with other high-level or low-level events. An event involves at least one participant (associated via the property `dul:hasParticipant`) and it holds for a specific `TimeInterval` specified by the property `dul:hasTimeInterval`.

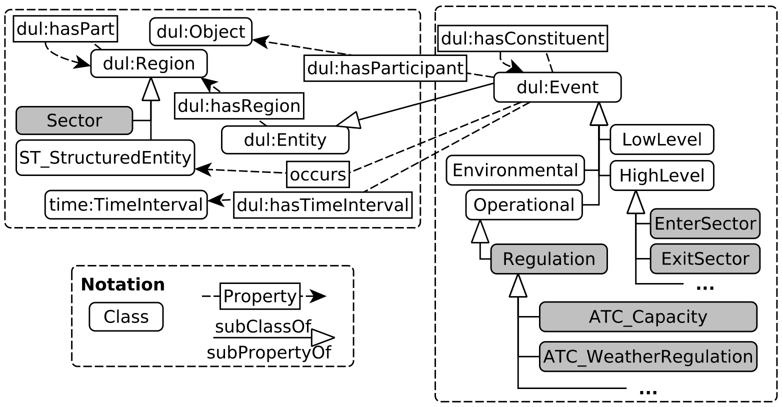


Figure 3: The pattern of trajectories linked with events. Domain specific concepts in gray

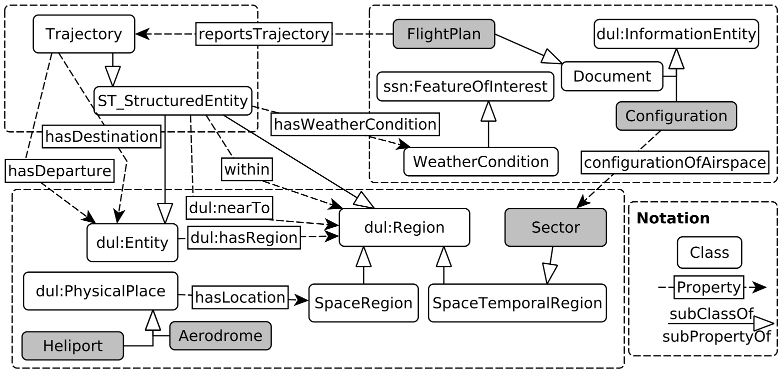


Figure 4: The pattern of trajectories linked with contextual information. Domain specific concepts in gray

It should be noted that associating events to trajectory parts satisfies the requirement to associate multiple events to varying levels of trajectory analysis, according to the information used for the detection of each event: For instance, a low-level “turn” event may co-occur with a low-level “descend” event and thus, both events may be associated to the same semantic node. In addition to that, this semantic node may be associated to a trajectory segment which in its own turn is associated with events of type “DescendingPhase” and “CrossingSector”.

In addition to events, trajectory parts of semantic trajectories can be linked to contextual information, i.e. information about entities in the environment that affect the moving object, including other trajectories. Such information may be archival information concerning static aspects of the environment (e.g. airports, airspaces, etc), dynamic (e.g. changing sector configurations), or streaming (e.g. weather forecasts). The pattern for linking trajectory parts with contextual information is illustrated in Figure 4.

Weather conditions are very important to trajectories in the MSA and ATM domains: Each `TrajectoryPart` can be associated with entities of type `WeatherCondition`, which is defined as a subclass of `ssn:FeatureOfInterest`, (i.e. the entity whose properties are being estimated or calculated in the course of an observation).

Of particular interest to the ATM and MSA domains are regions. Structured entities can be linked to spatial regions (instances of `dul:Region`) of particular interest through the properties `within` and `dul:nearTo`.

Also, the departure and destination of a trajectory can be considered as contextual information, linked via the properties `hasDeparture` and `hasDestination`, respectively. The properties range to the class `dul:PhysicalPlace`, which can be further refined to domain specific classes such as `Airport`, `Heliport`, or `Port`.

5. The SPARTAN Framework for Semantic Integration of Spatio-temporal Data

SPARTAN is a framework for semantic integration of streaming mobility data with other data sources. It comprises three main components, as illustrated in Figure 5: (a) Synopses Generator, (b) Data Transformation, and (c) Link Discovery. The components correspond to fundamental steps in the *big data analysis pipeline* [49], namely (a) data acquisition, cleaning, and filtering, (b) data extraction and representation, and (c) data integration.

In brief, streaming positional data, which is the primary data source of SPARTAN, is cleaned and compressed by the Synopses Generator. The output of the Synopses Generator along with other external data sources, streaming or archival, are transformed in RDF in accordance with the datAcron ontology, by

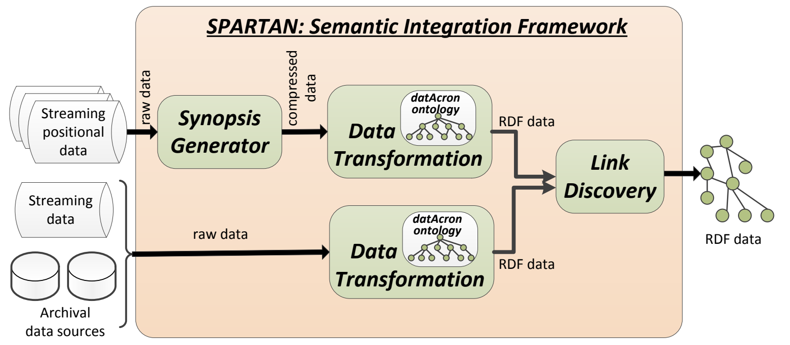


Figure 5: The SPARTAN framework for semantic integration of big spatio-temporal data.

the Data Transformation component. Then, the Link Discovery component can take as input any pair of "RDFized' sources to perform interlinking of entities and produce linked RDF data. Finally, SPARTAN outputs a stream of linked RDF data, which contains enriched trajectories of moving objects.

5.1. Trajectory Synopses Generator

Detecting important mobility events along trajectories has to carried out in a timely fashion against the streaming positional updates received from a large number of moving entities in our domains of interest. Instead of retaining every incoming position for each object, we consider trajectory synopses computed by a Synopses Generator module that drops any predictable positions along trajectory segments of "normal" motion characteristics. Indeed, except for adverse weather conditions, traffic regulations, local manoeuvres close to ports and airports, congestion situations, accidents, etc., most vessels and aircrafts normally follow almost straight, predictable routes at open sea and in the air, respectively. It turns out that a large amount of raw positional updates could be suppressed with minimal losses in accuracy, as they hardly contribute additional knowledge about their actual motion patterns. Instead of resorting to a costly trajectory simplification algorithm, we opt to reconstruct their traces approximately from judiciously chosen *critical points* along their trajectories. A critical point is represented by a *Semantic Node* in the datAcron ontology.

The Synopses Generator applies single-pass techniques for succinct, lightweight representation of trajectories without harming the quality of the resulting approximation. Effectively, we may keep only those positions conveying salient mobility features identified when the pattern of movement for a given object changes significantly. In particular, this online process can detect various types of critical points, as illustrated in Figure 6:

- *Stop* indicates that the object remains stationary (i.e., not moving) over a period of time. As this event has a duration but it must be detected at once, two critical points are emitted in order to specify its duration. *Start of stop* is identified for a just received location once its current instantaneous speed is lower than an appropriately chosen threshold for immobility (e.g., 0.5 knots). *End of stop* is identified once a previously stopped object starts to move again, i.e., its speed exceeds the threshold.

- *Slow motion* means that the object consistently moves at low speed (e.g., < 5 knots) over a period of time. The first and last of these positions will be reported as critical points, respectively annotated as *Start of slow motion* and *End of slow motion.*

- When the direction of movement has changed by more than a given angle, e.g., there is a difference of more than 5^o from its previous heading, this point is emitted as critical (*Change in Heading*).

- Along a trajectory, there may be a period when a change in the current instantaneous speed is under way. The respective critical point gets annotated as *Start of Speed Change.* Once the speed stabilizes again, the respective location is marked as *End of Speed Change.*

- *Communication gaps* may occur when a moving object has not emitted a message over a time period, e.g., the past 10 minutes. Hence, its course is unknown during this interval. Once communication is restored, the first location should be immediately emitted as a critical point (*Gap End*).

Reporting when this gap started may be important for safety reasons, so a notification is issued (*Gap Start*); in this latter case, this last-known position before the gap can only be annotated with delay.

- *Change in Altitude* may be detected for aircrafts by checking their *rate of climb* or *rate of descent*.
- *Takeoff* for aircrafts occurs when they go from the ground to flying in the air.
- *Landing* for flying aircrafts is the first reported location when they touch the ground.

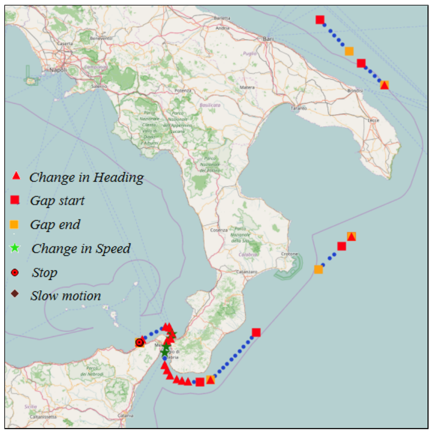
It is important that critical points should be emitted at *operational latency*, i.e., within milliseconds (or a few seconds at most) since the arrival of raw messages, so as not to cause delays in subsequent processing. In effect, once a location arrives in the incoming stream, it must be instantly characterized (ideally within milliseconds) if it qualifies as critical according to the aforementioned cases. Hence, this derived stream of *trajectory synopses* must keep in pace with the incoming raw streaming data so as to get incrementally annotated with semantically important mobility features once they get detected.

5.2. Data Transformation

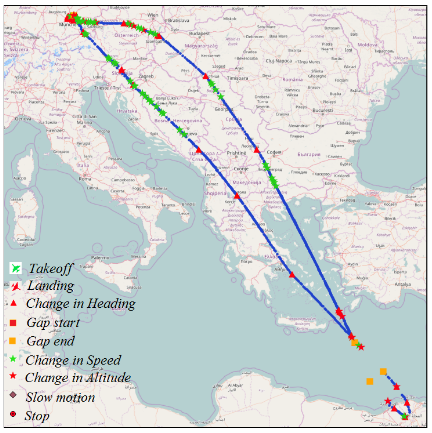
The purpose of *Data Transformation* is to extract all incoming data from its original form into RDF, in order to provide a basis for semantic integration. In essence, it performs the data extraction and data representation steps of the pipeline mentioned above. The Data Transformation is designed in a generic way and comprises two major components: a) the Data Connectors, and b) the Triple Generator.

5.2.1. Data Connectors

This component is responsible for connecting to a data source and providing access to its data. Its main task is to provide records of data from the data source to the core RDF generation component, which is the Triple Generator.



(a) Synopsis of a vessel trajectory



(b) Synopsis of an aircraft trajectory

Figure 6: Example synopses of critical points for (a) the maritime and (b) aviation use cases. Raw positions are shown as blue dots.

Data Connectors follow a *record-by-record* access model, treating both streaming and archival data sources in a uniform way: Essentially any data source is considered a "stream" of records that need to be processed with minimal latency. Since operations are performed on individual records, this results in minimizing the memory footprint of the RDF generation process, also providing opportunities for scalability and parallelization (even at the level of a single data source).

The current implementation includes a variety of data connectors that support data formats: (a) CSV files (surveillance, weather reports, registries of moving objects), (b) direct access to databases, (c) JSON messages, (d) XML files, (e) METAR/SPECI weather reports from offline or continuous feeds, (f) National Oceanic and Atmospheric Administration (NOAA) GRIB2 files for weather reports, (g) SPARQL endpoints, and (h) ESRI shapefiles.

5.2.2. Triple Generator

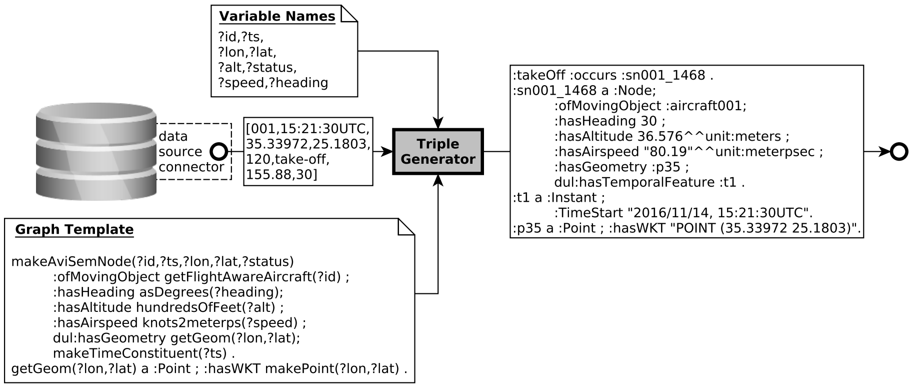
The Triple Generator is the core component of the RDF generation method. It converts each record provided by a data connector into a set of triples, i.e. a fragment of the RDF graph generated. This process uses two inputs: (a) a

Graph Template GT, and (b) a vector of variable names V. The generated triples are in accordance with the datAcron ontology. However, we note that the Triple Generator is a generic data transformation tool that can be parameterized with any given ontology, in order to transform raw input data to RDF.

The Graph Template inherits the idea of standard RDF Graph Patterns, i.e. it contains triples templates, where any of the three elements in an RDF triple (s,p,o) can be a variable or a custom function. The variables and the arguments of functions are identified by a leading question mark (?), e.g. ?imo, for the IMO[14] code of a vessel, may appear as a variable in a triple or as an argument of a function. For instance the triple template:

makeURI(?imo) rdf:type :Vessel .

uses the function makeURI(?imo), which constructs the URI of a resource for a given ?imo value. On the other hand, the vector V contains the variables that appear in the Graph Template, specifying mappings to fields in data sources.



```
?id,?ts,
?lon,?lat,
?alt,?status,
?speed,?heading
```

```
[001,15:21:30UTC,
35.33972,25.1803,
120,take-off,
155.88,30]
```

```
:takeOff :occurs :sn001_1468 .
:sn001_1468 a :Node;
        :ofMovingObject :aircraft001;
        :hasHeading 30 ;
        :hasAltitude 36.576^^unit:meters ;
        :hasAirspeed "80.19"^^unit:meterpsec ;
        :hasGeometry :p35 ;
        dul:hasTemporalFeature :t1 .
:t1 a :Instant ;
        :TimeStart "2016/11/14, 15:21:30UTC".
:p35 a :Point ; :hasWKT "POINT (35.33972 25.1803)".
```

```
makeAviSemNode(?id,?ts,?lon,?lat,?status)
        :ofMovingObject getFlightAwareAircraft(?id) ;
        :hasHeading asDegrees(?heading);
        :hasAltitude hundredsOfFeet(?alt) ;
        :hasAirspeed knots2meterps(?speed) ;
        dul:hasGeometry getGeom(?lon,?lat);
        makeTimeConstituent(?ts) .
getGeom(?lon,?lat) a :Point ; :hasWKT makePoint(?lon,?lat) .
```

Figure 7: Example of data transformation illustrating input, output, and configuration files.

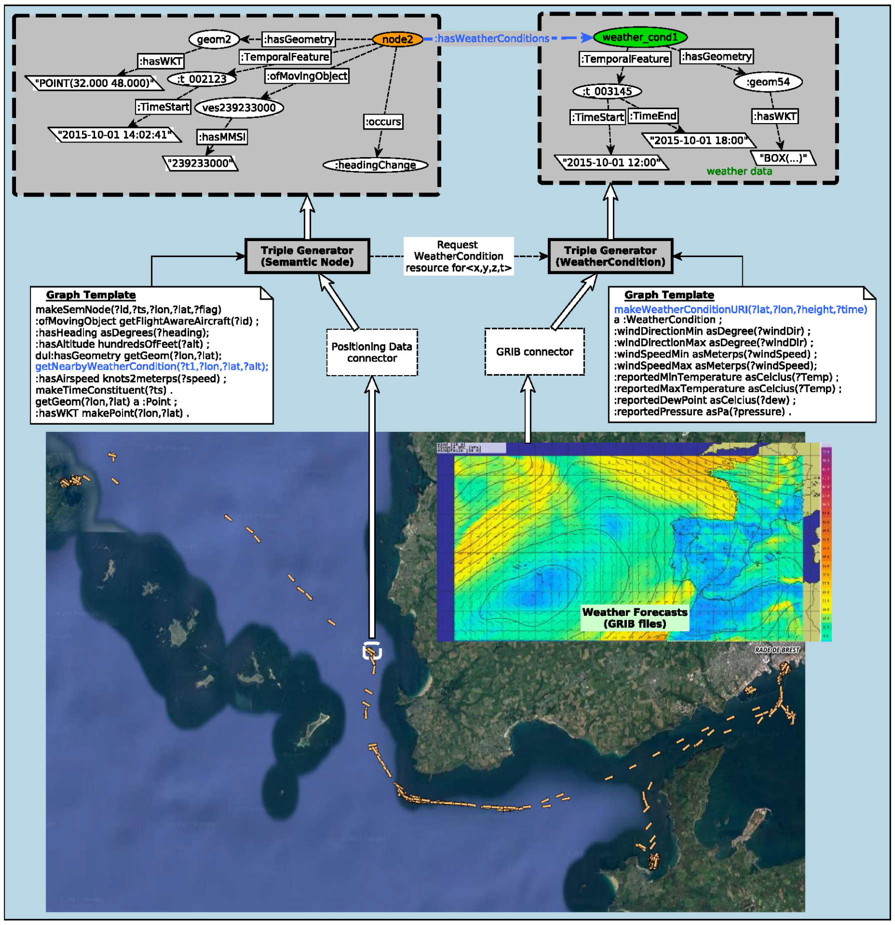
5.2.3. Examples of Data Transformation

A complete example of data transformation from a single data source to RDF with the corresponding configuration files is illustrated in Figure 7. The

[14] International Maritime Organization

data source in this example provides surveillance data from the aviation domain that describe the spatio-temporal position of moving objects (aircrafts). The input to the Triple Generator is a record provided by the appropriate data connector. A set of eight variables is provided in order to bind input values to variable names. For example, the first variable (?id) will be bind to the first value (column) of the input record (001), and refers to the aircraft ID. The graph template constructs trajectory (semantic) nodes for aircraft, given their ID their 3-D position (?lon, ?lat and ?alt for altitude), time point (?ts), status (?status), speed (?speed) and heading (?heading). The produced output follows the structure of the graph template, after having replaced variables with the respective input values, and functions with the result of the function call. Notice that the graph template is written in compliance with the datAcron ontology.

Figure 8 shows another example of data transformation, revealing the use of functions in triple patterns in processing and integrating multiple data sources. The positioning data connector provides surveillance data of vessels to a Triple Generator instance, similar to the previous example. The GRIB connector accesses binary files that contain weather forecasts and can extract the related weather attributes for a given spatio-temporal position. As soon as a new position is received, a request is made to the Triple Generator instance that retrieves the weather information and provides it in RDF representation based on the respective Graph Template. This allows data transformation of selected weather information, corresponding to the area defined by the positioning information. More interestingly, the weather Triple Generator returns the URI of the weather condition to the positioning Triple Generator, which can then create the `:hasWeatherCondition` property and associate the position with the weather. This is an example of lightweight, “close-to-the-sources” integration, which is also supported by our data transformation component.



Graph Template

```
makeSemNode(?id,?ts,?lon,?lat,?flag)
:ofMovingObject getFlightAwareAircraft(?id) ;
:hasHeading asDegrees(?heading);
:hasAltitude hundredsOfFeet(?alt) ;
dul:hasGeometry getGeom(?lon,?lat);
getNearbyWeatherCondition(?t1,?lon,?lat,?alt);
:hasAirspeed knots2meterps(?speed) ;
makeTimeConstituent(?ts) .
getGeom(?lon,?lat) a :Point ;
:hasWKT makePoint(?lon,?lat) .
```

Graph Template

```
makeWeatherConditionURI(?lat,?lon,?height,?time)
a :WeatherCondition ;
:windDirectionMin asDegree(?windDir) ;
:windDirectionMax asDegree(?windDir) ;
:windSpeedMin asMeterps(?windSpeed) ;
:windSpeedMax asMeterps(?windSpeed);
:reportedMinTemperature asCelcius(?Temp) ;
:reportedMaxTemperature asCelcius(?Temp) ;
:reportedDewPoint asCelcius(?dew) ;
:reportedPressure asPa(?pressure) .
```

Figure 8: Example of data transformation and close-to-the-sources integration.

5.3. Stream-based Spatio-temporal Link Discovery

The goal of the link discovery process is to identify the relations (links) between entities present in a *source* and *target* dataset. In our work, the majority of relations are of spatial or spatio-temporal nature, thus our focus is on spatio-temporal link discovery[15]. We develop a framework for spatio-

[15]Even though we focus on spatio-temporal relations, we can also compute other relations (e.g. sameAs) using spatial distance (e.g. sameAs relations between ports found in different datasets).

temporal link discovery which, by design, operates on streaming data sources, processing record-by-record (similar to the data connectors in Section 5.2) with low latency. Obviously, archival data sources are also supported, by enabling record-by-record access to the archival data. Core innovations of the proposed framework include the support of stream-based link discovery in combination with the provision of multiple spatio-temporal relations. Both these issues are not addressed by state-of-the-art link discovery methods, as explained in the Related Work section 2.4.

	Point-to-point	**Point-to-region**	**Region-to-region**
overlaps	undef.	undef.	✓
within	undef.	✓	✓
nearby	✓	✓	✓
nearest	✓	✓	✓

Table 1: Types of spatio-temporal relations supported by our framework.

Table 1 provides a summary of the types of spatio-temporal relations supported by our implementation (we have also implemented all OGC relations, such as cross, touch, equal, intersect, etc). "Point" refers to a spatio-temporal position, whereas "region" refers to a 2D spatial area described by a polygon or a 3D geometry. Point-to-point relations are of spatio-temporal nature, i.e., both the space and time are taken into account to determine whether two points are nearby, whereas region relations are of spatial nature. Supported relations include topological relations (*overlaps* and *within*) as well as proximity relations (*nearby* and *nearest*). Notice that state-of-the-art approaches for static spatial data, such as RADON [45], do not support proximity relations.

To provide meaningful examples of the afore-described relations in real-world situations, we briefly mention indicative cases in the following. For example, given the spatio-temporal position of a vessel, represented as a point $\{x, y, t\}$, *within* stands for enclosure of the point within a polygon representing some geographical area of interest. Other types of links correspond to spatial proximity relations, e.g. a vessel is located nearby a region, or spatio-temporal proximity relations, e.g. a vessel is near (or nearest) to another vessel. Notice that both spatial and spatio-temporal relations can be supported by our link discovery framework, which are not covered by existing link discovery frameworks [8].

5.3.1. Requirements for Spatio-temporal Link Discovery Framework

The design of our spatio-temporal link discovery framework is guided by the following main requirements:

- *Support for streaming sources:* the framework must inherently support streaming data sources in conjunction to archival ones, making no assumption on a priori access to the complete datasets;
- *Performance:* the framework must support efficient link discovery. To achieve this, different blocking techniques can be plugged in, tailored to the characteristics of the data source at hand, to achieve optimized performance;
- *Extensible set of relations:* the framework needs to allow easy addition of new distance/similarity functions, which enable the discovery of new spatio-temporal relations;
- *Generic nature:* the framework should not be tailored to a particular type of data source, but rather support various types of data sources and allow easy addition of a new data source.

Supporting even a subset of these requirements raises research challenges that go well beyond the state-of-the-art.

5.3.2. Architecture

The generic architecture of the implemented spatio-temporal link discovery framework is illustrated in Figure 9. Its inputs are a source and a target dataset with the corresponding connector components, which provide data as RDF graph fragments, in case this is not already the case. In addition, a configuration file for each dataset is provided as input, which specifies filtering conditions for the RDF triples that should be evaluated. In addition, it specifies the connection setup, the definition of the relation, the function used in the refinement process and the function used for accessing entities from the RDF graph fragment. In this way, the depicted *SourceConnector* and *TargetConnector* components filter RDF triples from each data source.

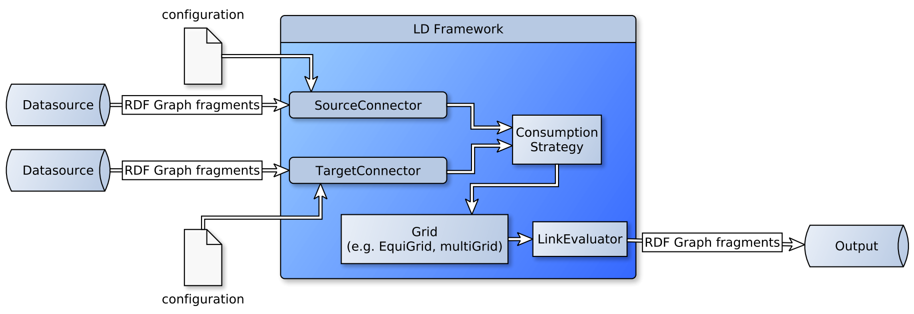


Figure 9: The architecture of spatio-temporal link discovery.

The core parts of the link discovery architecture are the following components:

- The *Consumption Strategy* specifies how input data should be processed. Currently, we have implemented two main strategies: (a) “target-first strategy” which proceeds to the source dataset only after the target data source has been accessed in its entirety, and (b) “joint strategy” which processes the two data sources and consumes them jointly. The former strategy is quite standard in link discovery frameworks which first access and organize the target dataset using some blocking technique, and then process the source dataset record-by-record. Instead, the latter strategy is quite different from strategies used by state-of-the-art link discovery frameworks, and is specifically tailored for streaming data sources. For example, it can be used to detect links between trajectory parts. Apparently, the target-first strategy enables tuning for better performance, such as swapping target-source data sources, in order to use blocking on the smaller dataset and reduce overall processing time (a technique used in [45]).

- The *Blocking Mechanism* in the case of spatio-temporal data is typically a grid structure that partitions the space in blocks. This allows effective filtering out large portions of data that cannot be linked to the currently examined object. Our current implementation provides two different types of grid: (a) EquiGrid, which given the granularity for each dimension, constructs equally sized cells, (b) Hierarchical grid, where multiple EquiGrids are employed for different granularities, thereby enabling more efficient pruning in some cases.

- The *Link Evaluator* comprises the functions that should be invoked to evaluate if a given relation (e.g., overlaps, within, nearby, etc.) between source and target entities exists. We have implemented the functions for all OGC relations between all combinations of static and moving entities, to detect the relations specified in Table 1. Finally, the linked entities detected by the link evaluator are directed to output. At this point, the framework provides the options to either: (a) concatenate the linked entities (as triples) to the corresponding RDF fragment used in the input, or (b) export only linked entities (as RDF triples). The first option provides synchronized and sequential RDF graph fragments to the output which is necessary in case of streaming source/target data sources, while the latter reduces the overall link discovery time and enables decoupling the processing of input datasets from the processing of discovered links.

6. The SPARTAN Big Data Architecture

In this section, we present the design of the SPARTAN big data architecture, focusing on the implementation of individual components as well as the communication mechanism used for integrating the different components.

6.1. Communication

SPARTAN follows a *loosely-coupled architecture*, which was a design decision that facilitates integration of components developed using different technologies. Moreover, our design makes the prototype easily extensible with other components. For the communication between components we opt for Apache Kafka.

Kafka[16] is a distributed, partitioned, replicated commit log service [50]. It provides the functionality of a messaging system. Kafka maintains feeds of messages in categories called *Topics*. Each Topic is partitioned for scalability and *Partitions* are distributed in the cluster. Processes that publish messages to a Kafka Topic are called *Producers*. Processes that subscribe to Topics and process the feed of published messages are called *Consumers*. Kafka runs in a *Cluster* comprised of one or more servers each of which is called a *Broker*.

In SPARTAN, all components take as input and provide as output a Kafka Topic. In this way, we achieve stream-based communication, while also enjoying scalability and fault-tolerance at the same time. To improve performance and provide scalability, Topics can be partitioned and then distributed to different machines in the cluster.

6.2. Implementation of Trajectory Synopses Generator

The Synopses Generator has been implemented on top of Apache Flink [51]. Currently, Apache Flink completely lacks support for *spatial and spatiotemporal entities* (e.g., points, lines, polygons, let alone moving objects with speed, heading, acceleration, etc.) and related *operations* (distance, speed, topological comparisons, etc.). Thus, we introduced custom data structures for maintaining critical points with support for all mobility operations and functions required in our processing flow. In addition, we have successfully defined in Avro [52] extensible *attribute schemata* for critical point types for the maritime and aviation use cases. These schemata are being used to hold all spatio-temporal properties from raw data, as well as those dynamically calculated during processing.

[16]`http://kafka.apache.org/`

Further, we have defined in Kafka [53] specific *topics* for all messages consumed and produced by the module, i.e., raw positional data as well as for derived critical points. All business logic in our methodology as detailed in Sections 5.1 has been implemented in Scala, with some auxiliary Java classes used for exposing the attribute schemata from Avro. Module implementation is separate for the maritime and aviation use cases for better maintenance and allowing independent execution of simulations with different configurations.

6.3. Implementation of Data Transformation

The Data Transformation component has been implemented in Java 8, including both Data Connectors and the Triple Generator. Its design to consume data in a record-by-record fashion, allows easy parallelization in different ways. In case of a high-rate streaming source, it is possible to have a single data connector that sends records to multiple instances of triple generator, thus matching the rate of the streaming source. In case of a vast-sized archival data source, is is possible to use multiple Data Connectors, each accessing a different range of underlying records, thus achieving parallelization at the level of data access. It should be noted that this is applicable for those data sources that are not constrained by sequential access to records. However, it is applicable for many data sources, including CSV and XML files, relational databases, etc.

Another noteworthy implementation aspect of the Data Transformation component is the one exemplified in Figure 8, offering "close-to-the-sources" integration. This example shows that it is possible to perform data transformation at two different sources and at the same time create associations between the generated RDF data. In technical terms, this is achieved by enabling communication between instances of Triple Generator.

6.4. Implementation of Link Discovery

As already mentioned, the implementation of the Link Discovery component adheres to a generic design, thus supporting alternative blocking mechanisms (spatial grids, R-tree, etc.) and allowing easy addition of new algorithms corresponding to new spatio-temporal relations. All currently implemented algorithms adhere to the filter-and-refine paradigm, where at the filtering step many combinations between source and target entities are pruned, and the remaining candidates are evaluated at the refinement step.

The spatio-temporal link discovery framework is applicable in a setting where one source is streaming and the other archival, as well as when both sources are streams. At the current stage, the implementation is centralized, however parallelization can be achieved by spatio-temporal partitioning techniques (tailored to the relation at hand), to ensure that each partition contains all necessary data to identify the correct links.

7. Experimental Evaluation

In this section, we first describe the main datasets used in the evaluation (Section 7.1). Then, we provide experimental results for the individual components in Section 7.2 in order to study their performance individually, and then present the empirical evaluation of the integrated prototype in Section 7.3.

7.1. Datasets

We use two maritime datasets in our evaluation study, denoted NARI (Institut de Recherche de l'École Navale) and IMISG (IMIS Global, a private company), based on the data provider. NARI contains AIS kinematic messages from vessels sailing in the Atlantic Ocean around the port of Brest, Brittany, France and span a period from 1 October 2015 to 31 March 2016. After deduplication of the original 19,035,630 AIS messages, this dataset yielded 18,495,677 point locations (kinematic AIS messages only), which was used as input at their original arrival rate for creating trajectory synopses. Attribute *MMSI* in the original records is used as the identifier for each of the 5055 vessels in this dataset. IMISG comes from AIS messages relayed from a very large fleet of 118,003 vessels in the Mediterranean Sea and part of the Atlantic Ocean during

January 2016. After decoding and deduplication of AIS messages, this dataset yielded 61,187,265 raw point locations (kinematic AIS messages only) that are being used in the simulations at their original arrival rate. Again, attribute *MMSI* in the original data is used as the identifier of each vessel.

7.2. Evaluation of Individual Components

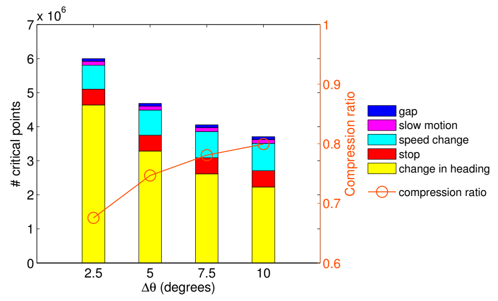
7.2.1. Trajectory Synopsis Generation

As every data reduction process, effectiveness of trajectory summarization is a trade-off between compression efficiency and approximation accuracy. Hence, regarding maintenance of *trajectory synopses*, we measured *compression ratio*. This is the percentage (%) of positions dropped from the approximate trajectory synopses over the raw ones originally obtained, or equivalently:

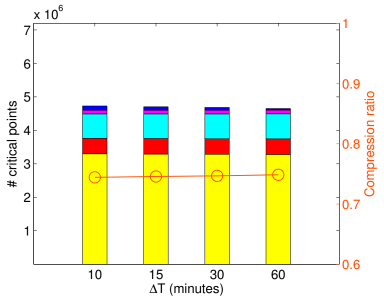
$$1 - \frac{\#critical\ points}{\#raw\ positions}.$$

The higher this ratio, the more compressed and lightweight the resulting synopses. A compression ratio closer to 1 signifies stronger data reduction, as the vast majority of original locations are dropped and few critical points suffice to represent the trajectories. The red line in the following plots depicts measurements of this ratio with varying parameters in order to quantify their effect on compression.

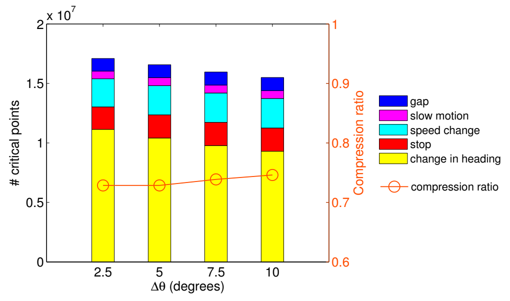
Experimental Results. For the two original maritime datasets, we examine compression ratio with respect to angle threshold $\Delta\theta$ and the minimum period ΔT for communication gaps. With a lower $\Delta\theta$, even slight deviations in vessel direction can be spotted, and thus extra critical points get issued. Bar charts in Figures 10(a) and 10(c) illustrate the amount of critical points in each class (gap, slow motion, speed change, stop, change in heading) retained from the entire dataset. Clearly, every further increase in threshold $\Delta\theta$ suppresses more and more turning points and does not affect the share of any other class of critical points, but incurs extra reduction in the total amount of emitted critical points. Compression ratio always remains above 70%, and with a more relaxed $\Delta\theta$ it



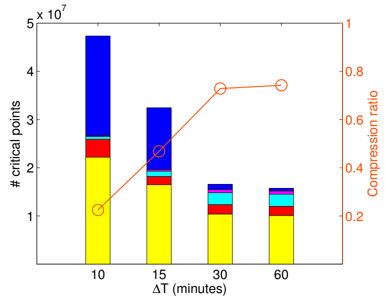
(a) *varying angle on NARI*



(b) *varying gap period on NARI*



(c) *varying angle on IMISG*



(d) *varying gap period on IMISG*

Figure 10: Effect of parametrization on compression ratio for *original maritime* datasets

reaches as much as 80% for the NARI dataset. In this latter case, only 20% of the original locations survive as critical, mostly by eliminating local manoeuvres of little impact on vessel's course. Eliminating noise also plays an important role in data reduction, as erroneous deviations are dropped and no points need be retained. Data reduction over the IMISG dataset is less intense, since contents of this decimated stream have been already downsampled, apparently leaving very small margin for further compression. Note that even allowing an angle tolerance of 10^o for detecting turning points incurs very little extra reduction compared to the same test on the NARI dataset.

A similar pattern regarding reduction efficiency can be observed in Figures 10(b) and 10(d) with respect to varying periods ΔT for detecting gaps in communication. Not surprisingly, it is the amount of critical points marking those gap periods that gets reduced with an increasing threshold ΔT. For the

NARI dataset, reduction ratio is never below 76%, and only a tiny percentage of critical points concern communication gaps greater than 10 minutes, as most the fishing boats in this fleet report frequently enough their position. This is in contrast with the IMISG dataset (Figure 10(d)), where almost half of the detected critical points for $\Delta T = 10$ minutes mark such gaps and compression ratio drops to 23%, practically retaining three out of four raw points in the synopsis. Indeed, keeping track that contact was lost even for 10 minutes for a given vessel incurs a surge in the amount of such critical points in the synopses. This is another side effect of the poor reporting frequency observed in this dataset and its severe impact on effective trajectory detection. Of course, relaxing the gap threshold can substantially increase compression ratio up to 74%. It is no wonder that reduction practically stabilizes above ΔT= 30 minutes, as the average reporting frequency in the IMISG dataset is about every 20 minutes.

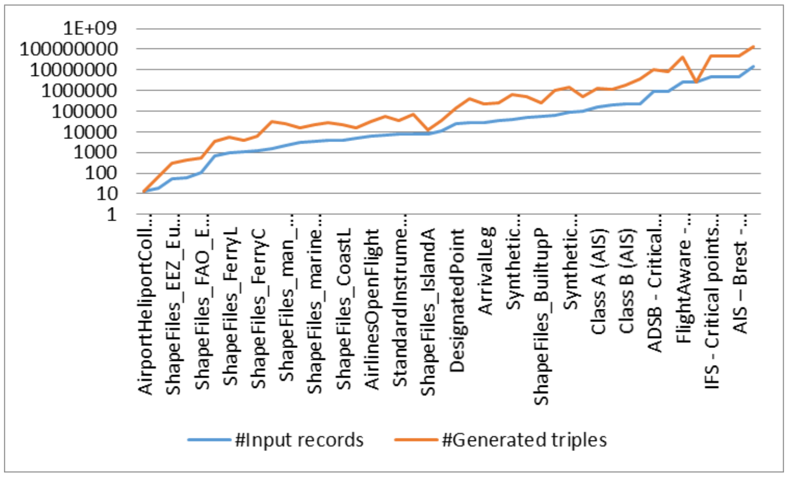


Figure 11: Number of input records (blue line) and generated triples (red line) per data source, ordered by number of input records in the data source.

7.2.2. Data Transformation to RDF

Figures 11 and 12 provide evaluation results from the RDF generation process for various real-world data sources (including IMISG and NARI), both from

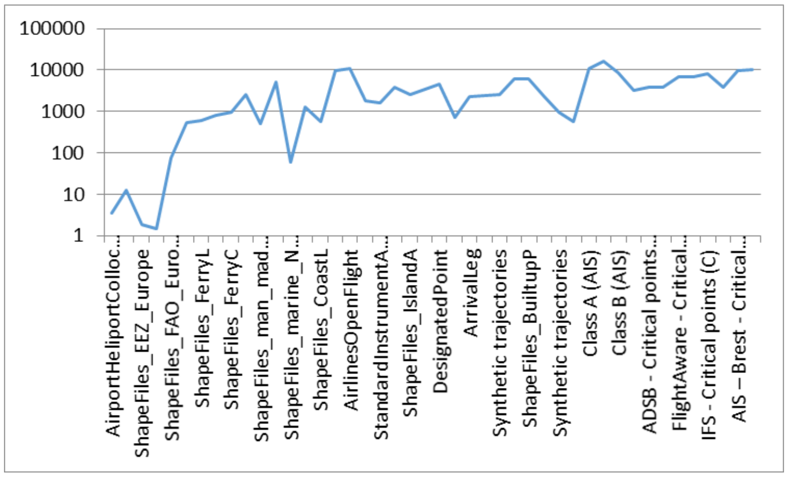


Figure 12: Number of generated triples per second for each data source, ordered by number of input records in the data source.

the maritime and aviation domains. In both figures, the X-axis indicates the data source, ordered by the volume of input data. More than 40 data sources have been included in the study, having heterogeneous formats, varying size, and complexity.

Figure 11 depicts the number of raw records and the corresponding number of generated triples per data source. Notice the log scale on the y-axis. The number of generated triples is larger than the raw records, since typically each attribute of a single record is mapped to a triple. It must be pointed out that for a surveillance data source (denoted AIS Brest and corresponding to the summarized NARI dataset mentioned above) we generate 131M triples.

In Figure 12, we evaluate the performance of the data transformation process for these data sources. The chart depicts the number of processed input records per second for the different data sources including the "close-to-the-sources" integration of surveillance with weather data, again using log scale on the y-axis. Focusing again on maritime surveillance data, we managed to transform 10,467 input records to RDF per second. For some sources this number is smaller due to complicated geometries that are computationally intensive or complex XML-based formats (e.g., AIXM) that require navigation of multiple paths to retrieve the necessary information. Overall, the average time per triple generated is approximately 0.04 seconds, given that the frequency of position reporting per aircraft/vessel is at least 2 seconds. This experiment clearly demonstrates the efficiency of the proposed RDF generation method.

7.2.3. Spatio-temporal Link Discovery

In this section we provide experimental results for two of the relations mentioned in Table 1: *within* and *nearby.* In both cases, the source stream of the critical points produced by the Synopses Generator takes as input the IMISG dataset. The number of critical points is 14,314,312. For the target dataset, we employ: (a) for relation *within*, the "Natura2000" dataset which consists of 3,552 polygons representing protected areas, and (b) for relation *nearby*, the "Natural Earth Data Beacons" dataset, which consist of 7,942 entities.

In both cases, we use an EquiGrid as a blocking mechanism to organize the target dataset in memory, and then process each position of the IMISG dataset in a streaming fashion, in order to discover whether one of the above relations exists and perform the linking. The granularity of the grid was set to 1.5x2.5 degrees. Obviously, in some cases, a polygon is assigned to multiple grid cells, depending on its position.

It should be noted that the link discovery method is 100% accurate, i.e., is always discovers the correct links/relations. Therefore, we turn our attention to evaluating the performance. Our main metric is the number of IMISG positions processed per second (i.e., throughput), which reflects the efficiency of link discovery. In addition, we measure the *ComparisonsRatio*, which is the fraction of comparisons made over the theoretical number of comparisons necessary:

$$ComparisonsRatio = \frac{FalsePositive + TruePositive}{TotalEntitiesInSource \times TotalEntitiesInTarget}$$

where *FalsePositive* is the number of comparisons that did not produce a link and *TruePositive* is the number of comparisons that actually produced a link.

Table 2 reports the results of the experimental evaluation. The throughput is 773.69 positions/sec for relation *nearby* and 647.94 positions/sec for *within.*

Relation	**Source/Target**	**Throughput**	**ComparisonsRatio**
Nearby	IMISG/Beacons	773.69	1.4%
Within	IMISG/Natura2000	647.94	0.39%

Table 2: Performance of link discovery for relations *within* and *nearby*.

Thus, we are able to discover links for several hundreds of positions per second. Notice that these results are based on a single-threaded implementation, and the task can be naturally parallelized. The *ComparisonsRatio* is also quite small, in the order of 1%, showing that the use of the grid drastically reduces the number of required comparisons for discovering the links.

7.3. Evaluation of Architecture

In this section, we provide experimental results of the SPARTAN architecture based on a deployment in a medium-sized 10-node cluster at the University of Piraeus. Each node is equipped with: Intel Xeon E5 (6 cores, 1.6GHz), 128GB DDR4 RAM, 6TB HDD + 200GB SSD, and 1Gbit connection between each node. The software stack of our cluster is: Ubuntu: 16.04.2 (kernel 4.4.0) x64, Java: 1.8.0_121, Hadoop, HDFS, YARN: 2.7.2, Scala: 2.11.7, Flink: 0.10.2, and Confluent: 3.1.1. Our Kafka cluster consists of three brokers, each allotted with 16GB of memory.

The evaluation is performed using two of the three SPARTAN components, namely the Data Transformation and the Link Discovery, since Synopses Generator is real-time and does not incur any further delay to streams. We varied the following parameters:

- Replication factor: in our experimental study we used replication factor 2 and 3. A higher replication factor increases durability and achieves higher availability.
- Use of callbacks for sends: The use of callbacks allows sends to be asynchronous, meaning that we do not have to wait for an acknowledgment from the broker, before sending the next message.

We opted to measure throughput in number of TTL[17] records per second. In our experiment, a TTL record typically corresponds to 10 RDF triples.

The throughput of the integrated prototype was 2,256 TTL records per second for replication factor 3, and only slightly lower (2,252) for replication factor 2, with callbacks enabled. This experiment demonstrates that SPARTAN is able to provide enriched streams of generated RDF data with high rate and low latency. It should be mentioned that in our applications (maritime and aviation) the surveillance streams are of much lower rate. We also measured the throughput of each component separately, and the data transformation component achieved 3,408 and 3,708 TTL records per second. This shows that the link discovery is the slowest component between the two, and it is responsible for the overall throughput achieved. When callbacks were not used, the throughput of the data transformation component increased to 6,000 TTL records per second. All in all, our evaluation shows that the integrated prototype achieves high throughput. However, we believe that improvements can be achieved by careful tuning of other Kafka-related parameters, and this is mostly an engineering issue that deserves further study.

8. Discussion

Following SPARTAN's way, raw trajectory data is transformed into multidimensional sequences (semantic trajectory data) that form a more realistic representation model of the complex every-day life [1]; mobility of vessels belongs to this broad class. Operating on such compressed but semantified time-series may facilitate several analysis tasks. For instance, clustering analysis may benefit from additional variables by incorporating the principle of divide-and-conquer via a semantic-aware clustering of the routes, essentially grouping together routes that exhibit similarity not just in their spatio-temporal path

[17]Terse RDF Triple Language: a textual syntax for RDF that allows RDF graphs to be completely written in a compact and natural text form

but also in their semantic vectors. The complexity of having additional semantic dimensions over the spatio-temporal domain can be addressed efficiently by employing a proper similarity function, which is typically some linear vector norm. This enables the incorporation of any number of semantic parameters to be used, e.g. weather conditions, moving object properties, etc, with limited impact on the overall complexity. Moreover, the clustering may be performed using a properly designed semantic-aware similarity function that takes into account the entire input space instead of just the spatio-temporal proximity of the trajectories. In other words, we may follow an approach that addresses the fact of routes that may seem identical in the spatio-temporal domain but may be further grouped due to different weather conditions. This is extremely important for creating clusters that are compact, not only in the spatio-temporal domain (geodesic proximity) but in the full semantic-enriched domain.

Furthermore, predictive analytics may directly benefit from such fine-grained distinction of patterns, but they may further take advantage of the additional enriched variables (features) by training additional predictive models on the deviation of these features to predetermined values, i.e., the intended itinerary of a vessel along, as well as other "enrichment" parameters such as localized weather and vessel properties. Preliminary results, such as those reported in [54], indicate that trajectory clustering and prediction can be improved when the underlying positioning data are augmented with weather conditions.

In the case of complex events recognition, a typical use-case concerns detecting specific patterns (sequences of low-level events). For example, a sequence of two low-level events is when a moving object *enters* and *stays within* a prohibited area for some temporal period, in which case an alert must be issued. By associating the position of a moving object with the geometrical representations of protected areas in real-time, SPARTAN enriches trajectories with such low-level spatio-temporal events that facilitate complex event recognition.

9. Conclusions and Outlook

In this paper, we presented SPARTAN, a framework for real-time semantic integration of big mobility data with other data sources, aiming at providing enriched trajectories that are exploited by higher level analysis tasks. Our framework contains methods for data cleaning and filtering, data transformation, and link discovery, thereby offering an end-to-end solution to the problem of providing enriched streams of mobility data. In our future work, we intend to study in depth how the enriched data can improve the quality of different data analysis tasks, such as trajectory prediction and trajectory clustering.

Acknowledgements

This work is supported by the datAcron project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 687591 (`http://datacron-project.eu`).

References

[1] C. Parent, S. Spaccapietra, C. Renso, G. L. Andrienko, N. V. Andrienko, V. Bogorny, M. L. Damiani, A. Gkoulalas-Divanis, J. A. F. de Macêdo, N. Pelekis, Y. Theodoridis, Z. Yan, Semantic trajectories modeling and analysis, ACM Comput. Surv. 45 (4) (2013) 42.

[2] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, K. Aberer, Semantic trajectories: Mobility data computation and annotation, ACM TIST 4 (3) (2013) 49.

[3] C. Claramunt, C. Ray, E. Camossi, A. Jousselme, M. Hadzagic, G. L. Andrienko, N. V. Andrienko, Y. Theodoridis, G. A. Vouros, L. Salmon, Maritime data integration and analysis: recent progress and research challenges, in: Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017., 2017, pp. 192–197.

[4] K. Patroumpas, A. Artikis, N. Katzouris, M. Vodas, Y. Theodoridis, N. Pelekis, Event recognition for maritime surveillance, in: Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, 2015, pp. 629–640.

[5] K. Patroumpas, E. Alevizos, A. Artikis, M. Vodas, N. Pelekis, Y. Theodoridis, Online event recognition from moving vessel trajectories, GeoInformatica 21 (2) (2017) 389–427.

[6] A. N. Ngomo, S. Auer, LIMES - A time-efficient approach for large-scale link discovery on the web of data, in: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011, 2011, pp. 2312–2317.

[7] R. Isele, A. Jentzsch, C. Bizer, Efficient multidimensional blocking for link discovery without losing recall, in: Proceedings of the 14th International Workshop on the Web and Databases 2011, WebDB 2011, Athens, Greece, June 12, 2011, 2011.

[8] M. Nentwig, M. Hartung, A. N. Ngomo, E. Rahm, A survey of current link discovery frameworks, Semantic Web 8 (3) (2017) 419–436.

[9] G. Santipantakis, G. Vouros, A. Glenis, C. Doulkeridis, A. Vlachou, The datAcron ontology for semantic trajectories, in: ESWC-Poster Session, 2017.

[10] G. Santipantakis, G. Vouros, C. Doulkeridis, A. Vlachou, G. Andrienko, N. Andrienko, G. Fuchs, J. M. C. Garcia, M. G. Martinez, Specification of semantic trajectories supporting data transformations for analytics: The datAcron ontology, in: Proceedings of Semantics, 2017.

[11] D. L. Phuoc, M. Dao-Tran, J. X. Parreira, M. Hauswirth, A native and adaptive approach for unified processing of linked streams and linked data, in: The Semantic Web - ISWC 2011 - 10th International Semantic Web

Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I, 2011, pp. 370–388.

[12] E. Kharlamov, T. Mailis, G. Mehdi, C. Neuenstadt, Ö. L. Özçep, M. Roshchin, N. Solomakhina, A. Soylu, C. Svingos, S. Brandt, M. Giese, Y. E. Ioannidis, S. Lamparter, R. Möller, Y. Kotidis, A. Waaler, Semantic access to streaming and static data at siemens, J. Web Sem. 44 (2017) 54–74.

[13] J. Calbimonte, Ó. Corcho, A. J. G. Gray, Enabling ontology-based access to streaming data sources, in: The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I, 2010, pp. 96–111.

[14] E. Kharlamov, Y. Kotidis, T. Mailis, C. Neuenstadt, C. Nikolaou, Ö. L. Özçep, C. Svingos, D. Zheleznyakov, S. Brandt, I. Horrocks, Y. E. Ioannidis, S. Lamparter, R. Möller, Towards analytics aware ontology based access to static and streaming data, in: The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part II, 2016, pp. 344–362.

[15] E. Kharlamov, S. Brandt, E. Jiménez-Ruiz, Y. Kotidis, S. Lamparter, T. Mailis, C. Neuenstadt, Ö. L. Özçep, C. Pinkel, C. Svingos, D. Zheleznyakov, I. Horrocks, Y. E. Ioannidis, R. Möller, Ontology-based integration of streaming and static relational data with optique, in: Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016, 2016, pp. 2109–2112.

[16] D. L. Phuoc, H. N. M. Quoc, H. N. Quoc, T. T. Nhat, M. Hauswirth, The graph of things: A step towards the live knowledge graph of connected things, J. Web Sem. 37-38 (2016) 25–35.

[17] D. F. Barbieri, D. Braga, S. Ceri, E. D. Valle, M. Grossniklaus, C-SPARQL:

a continuous query language for RDF data streams, Int. J. Semantic Computing 4 (1) (2010) 3–25.

[18] D. Dell'Aglio, J. Calbimonte, E. D. Valle, Ó. Corcho, Towards a unified language for RDF stream query processing, in: The Semantic Web: ESWC 2015 Satellite Events - ESWC 2015 Satellite Events Portorož, Slovenia, May 31 - June 4, 2015, Revised Selected Papers, 2015, pp. 353–363.

[19] L. O. Alvares, V. Bogorny, B. Kuijpers, J. A. F. de Macêdo, B. Moelans, A. A. Vaisman, A model for enriching trajectories with semantic geographical information, in: GIS, 2007, p. 22.

[20] M. Baglioni, J. A. F. de Macêdo, C. Renso, R. Trasarti, M. Wachowicz, Towards semantic interpretation of movement behavior, in: Advances in GIScience, Springer, 2009, pp. 271–288.

[21] V. Bogorny, C. Renso, A. R. de Aquino, F. de Lucca Siqueira, L. O. Alvares, Constant - A conceptual data model for semantic trajectories of moving objects, Trans. GIS 18 (1) (2014) 66–88.

[22] T. P. Nogueira, H. Martin, Querying semantic trajectory episodes, in: Proc. of MobiGIS, 2015, pp. 23–30.

[23] R. Fileto, C. May, C. Renso, N. Pelekis, D. Klein, Y. Theodoridis, The baquara^2 knowledge-based framework for semantic enrichment and analysis of movement data, Data Knowl. Eng. 98 (2015) 104–122.

[24] D. Douglas, T. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, The Canadian Cartographer 10 (2) (1973) 112–122.

[25] C. Long, R. C.-W. Wong, H. V. Jagadish, Trajectory simplification: On minimizing the direction-based error, Proc. VLDB Endow. 8 (1) (2014) 49–60.

[26] J. Muckell, P. W. O. Jr., J.-H. Hwang, C. Lawson, S. S. Ravi, Compression of trajectory data: A comprehensive evaluation and new approach, Geoinformatica 18 (3) (2014) 435–460.

[27] H. Cao, O. Wolfson, G. Trajcevski, Spatio-temporal data reduction with deterministic error bounds, VLDB Journal 15 (3) (2006) 211–228.

[28] N. Meratnia, R. de By, Spatiotemporal compression techniques for moving point objects, in: EDBT, 2004, pp. 765–782.

[29] M. Potamias, K. Patroumpas, T. Sellis, Sampling trajectory streams with spatiotemporal criteria, in: Proceedings of the 18th International Conference on Scientific and Statistical Database Management, SSDBM 2006, 2006, pp. 275–284.

[30] P. Cudré-Mauroux, E. Wu, S. Madden, Trajstore: An adaptive storage system for very large trajectory data sets, in: Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, 2010, pp. 109–120.

[31] O. Wolfson, A. Sistla, S. Chamberlain, Y. Yesha, Updating and querying databases that track mobile units, Distributed & Parallel Databases 7 (3) (1999) 257–287.

[32] R. Lange, F. Dürr, K. Rothermel, Efficient real-time trajectory tracking, VLDB Journal 20 (5) (2011) 671–694.

[33] J. Liu, K. Zhao, P. Sommer, S. Shang, B. Kusy, R. Jurdak, Bounded quadrant system: Error-bounded trajectory compression on the go, in: 31st IEEE International Conference on Data Engineering, ICDE 2015, 2015, pp. 987–998.

[34] X. Lin, S. Ma, H. Zhang, T. Wo, J. Huai, One-pass error bounded trajectory simplification, PVLDB 10 (7) (2017) 841–852.

[35] S. Capadisli, S. Auer, A. N. Ngomo, Linked SDMX data: Path to high fidelity statistical linked data, Semantic Web 6 (2) (2015) 105–112.

[36] G. Santipantakis, G. Vouros, A. Glenis, C. Doulkeridis, A. Vlachou, Generating linked RDF data from heterogeneous streaming and archival data sources: Populating the datAcron ontology, in: Semantics-Poster Session, 2017.

[37] M. Hert, G. Reif, H. C. Gall, A comparison of rdb-to-rdf mapping languages, in: Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011, 2011, pp. 25–32.

[38] A. Dimou, M. V. Sande, P. Colpaert, R. Verborgh, E. Mannens, R. V. de Walle, RML: A generic language for integrated RDF mappings of heterogeneous data, in: Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014., 2014.

[39] F. Scharffe, G. Atemezing, R. Troncy, F. Gandon, S. Villata, B. Bucher, F. Hamdi, L. Bihanic, G. Képéklian, F. Cotton, J. Euzenat, Z. Fan, P.-Y. Vandenbussche, B. Vatant, Enabling linked data publication with the datalift platform, 2012.

[40] M. Lefrançois, A. Zimmermann, N. Bakerally, A SPARQL extension for generating RDF from heterogeneous formats, in: Proc. Extended Semantic Web Conference (ESWC'17), 2017.

[41] A. N. Ngomo, Link discovery with guaranteed reduction ratio in affine spaces with minkowski measures, in: The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I, 2012, pp. 378–393.

[42] A. N. Ngomo, A time-efficient hybrid approach to link discovery, in: Pro-

ceedings of the 6th International Workshop on Ontology Matching, Bonn, Germany, October 24, 2011, 2011.

[43] P. Smeros, M. Koubarakis, Discovering spatial and temporal links among RDF data, in: Proceedings of the Workshop on Linked Data on the Web, LDOW 2016, co-located with 25th International World Wide Web Conference (WWW 2016), 2016.

[44] A. N. Ngomo, ORCHID - reduction-ratio-optimal computation of geo-spatial distances for link discovery, in: The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I, 2013, pp. 395–410.

[45] M. A. Sherif, K. Dreßler, P. Smeros, A. N. Ngomo, Radon - rapid discovery of topological relations, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA., 2017, pp. 175–181.

[46] S. Moore, Benefits of highly predictable flight trajectories in performing routine optimized profile descents: Current and recommended research, in: Environmental Working Group Operations Standing Committee 2009 Annual Workshop, NASA Ames, 2009.

[47] R. Reisman, J. Murphy, P. Krolak, R. Wright, Modeling tactical trajectory accuracy effects on traffic flow management operations, in: AIAA Aviation Tech., Integration and Operations Conf, 2007.

[48] G. A. Vouros, A. Vlachou, G. M. Santipantakis, C. Doulkeridis, N. Pelekis, H. V. Georgiou, Y. Theodoridis, K. Patroumpas, E. Alevizos, A. Artikis, C. Claramunt, C. Ray, D. Scarlatti, G. Fuchs, G. L. Andrienko, N. V. Andrienko, M. Mock, E. Camossi, A. Jousselme, J. M. C. Garcia, Big data analytics for time critical mobility forecasting: Recent progress and research challenges, in: Proceedings of the 21th International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26-29, 2018., 2018, pp. 612–623.

[49] H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, C. Shahabi, Big data and its technical challenges, Commun. ACM 57 (7) (2014) 86–94.

[50] G. Wang, J. Koshy, S. Subramanian, K. Paramasivam, M. Zadeh, N. Narkhede, J. Rao, J. Kreps, J. Stein, Building a replicated logging system with Apache Kafka, PVLDB 8 (12) (2015) 1654–1655.

[51] Apache Flink, `https://flink.apache.org/`.

[52] Apache Avro data serialization system, `https://avro.apache.org/docs/current/`.

[53] Apache Kafka distributed streaming platform, `https://kafka.apache.org/`.

[54] S. Ayhan, H. Samet, Time series clustering of weather observations in predicting climb phase of aircraft trajectories, in: Proceedings of the 9th ACM SIGSPATIAL International Workshop on Computational Transportation Science, IWCTS@SIGSPATIAL 2016, Burlingame, California, USA, October 31 - November 3, 2016, 2016, pp. 25–30.