Spatio-Temporal-Keyword Pattern Queries over Semantic Trajectories with Hermes@Neo4j

Fragkiskos Gryllakis
Dept. of Informatics
University of Piraeus
Piraeus, Greece
fgryllakis@unipi.gr

Nikos Pelekis
Dept. of Statistics and Ins. Science
University of Piraeus
Piraeus, Greece
npelekis@unipi.gr

Christos Doulkeridis
Dept. of Digital Systems
University of Piraeus
Piraeus, Greece
cdoulk@unipi.gr

Stylianos Sideridis
Dept. of Informatics
University of Piraeus
Piraeus, Greece
ssider@unipi.gr

Yannis Theodoridis
Dept. of Informatics
University of Piraeus
Piraeus, Greece
ytheod@unipi.gr

ABSTRACT

In this paper, we demonstrate Hermes@Neo4j[1], an extension of Neo4j graph DMBS for semantic trajectories of moving objects, on the so-called Spatio-Temporal-Keyword Pattern queries. For this purpose, our engine exploits on hybrid Spatio-Temporal-Keyword (STK) index structures, also boosted by an appropriate selectivity estimation model. Hermes@Neo4j functionality is demonstrated over synthetic and real semantic trajectory datasets.

1 INTRODUCTION

The efficient management and analysis of the spatio-temporal evolution of a moving object (the so-called object's trajectory) has led to the development of plenty of appropriate index structures and algorithms, and even extensions over DBMS during the last two decades [1-5]. Recently, the research community has turned its interest to semantic trajectories [6], where spatio-temporal information is enriched with related annotations about the what, how, and why of movement [6-8].

The paradigm of Location-based Social Networking (LBSN) services, such as Twitter, Instagram and Foursquare, is indicative of this shift: the management and analytics over large amounts of spatio-temporal-textual data may result in useful conclusions about the users' behaviour.

Our motivation in this work is to demonstrate how a Semantic Trajectory Database (STD), built on top of an extensible DBMS, can efficiently support queries where constraints are set over the triple (spatial, temporal, textual) nature of semantic trajectories. In particular, we aim to demonstrate the functionality of our Hermes@Neo4j STD engine over Spatio-Temporal-Keyword Pattern (STKP) queries [9].

According to [9], an STKP query is defined as follows: Let E_i be a semantic trajectory episode abstraction, which is defined as a partially or completely defined episode. An episode abstraction therefore is an episode where some of its properties – spatial, temporal, textual information – may be missing. An STKP query over a STD takes as input a sequence Q of episode abstractions or the * wildchar (more formally, Q := <p* | p is either an episode abstraction E_i or the * wildchar>) and gives as output the semantic trajectories in STD that are compatible with Q.

An example of STKP query follows in Fig. 1 [9].

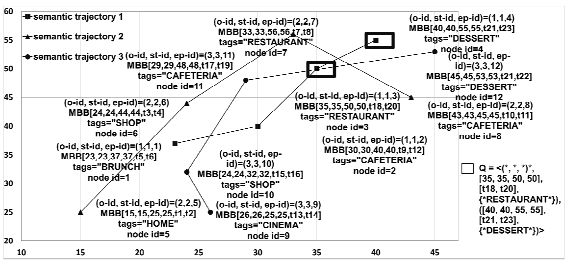

Figure 1: Graph representation of an STD consisting of 3 trajectories along with a STKP query.

In Fig. 1, we depict a STD consisting of 3 semantic trajectories; each trajectory consists of four episodes. An example STKP query Q is also illustrated at the bottom right corner. In particular, Q consists of a number of episode abstractions; with notation E_i* corresponding to a number of

[1]More information regarding Hermes@Oracle and Hermes@Postgres are available at www.datastories.org/hermes.

zero or more episode abstractions of the form E_i. For clarity of presentation the episode abstractions in Q distinguish the temporal from the spatial information, which is not the case in reality where both are organized together in a Minimum Bounding Box (MBB). Actually, Q searches for trajectories starting with zero or more episodes of any kind (see notation (*, *, *)* in Q), followed by an episode in a spatial [35, 35, 50, 50] and temporal [t_{18}, t_{20}] range with keyword 'RESTAURANT' and ending with an episode in a spatial [40, 40, 55, 55] and temporal [t_{21}, t_{23}] range with keyword 'DESSERT'. The output set includes semantic trajectory 1, which fulfills the above constraints.

The practical contribution of this work is that we present a framework that utilizes recently introduced (a) state-of-the-art hybrid indexes and (b) query processing algorithms on (c) a new STD engine, coined Hermes@Neo4j STD engine.

Hermes@Neo4j STD engine provides efficient and effective storage, indexing mechanisms and a library of utilities that facilitate spatio-temporal and textual operations on data, able to support STDs. More specifically, the merits and contributions of Hermes@Neo4j STD engine are summarized below:

1. Following the successful MOD engine paradigm of Hermes@Oracle [7,8], we designed a new datatype system for the representation and management of Semantic Trajectories into the extensible DBMS architecture of Neo4j [10], an ACID-compliant transactional NoSQL DBMS with native graph storage and processing, implemented in Java. The datatype system is formulated in the context of the graph database, that provides an intuitive model in our case.
2. Neo4j Spatial library [11], which is a library of utilities for Neo4j that facilitates spatial operations on data, is extended for processing the spatio-temporal and textual information of semantic trajectories.
3. We designed efficient access methods for semantic trajectories, called TSR-tree and TSR-tree indexes, for the hybrid indexing of both the spatio-temporal and the textual component. The hybrid indexes combine a spatiotemporal and text index tightly, such that both types of information can be used to prune the search space simultaneously during the spatiotemporal keyword query algorithm processing in STDs.
4. We developed efficient query processing algorithms upon the proposed indices in order to support a useful query type at the STD level, called STKP, as well as algorithms for efficiently importing semantic trajectories into STDs.

Employing separate indexes is weaker in comparison with the tightly integrated proposed state-of-the-art approach, since an efficient resolution of the STKP requires repetitive invocation of spatio-temporal-keyword matching queries.

The paper is organized as follows: in Section 2 we provide a brief presentation of the system architecture, the underlying indexes, etc. (the interested reader is referred to [9] for more details). Section 3 provides more information about system implementation details. Finally, Section 4 outlines the flow of the demonstration.

2 SYSTEM ARCHITECTURE

In this section, we present the core information about the architecture of our framework, illustrated in Fig. 2.

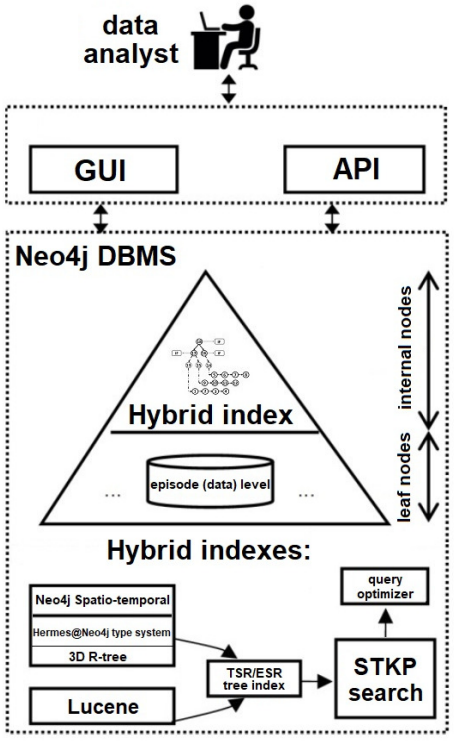

Figure 2: Hermes@Neo4j STD engine architecture.

2.1 Hybrid indexes

Neo4j Spatial R-tree index is extended to support spatio-temporal and classical trajectory-based queries (3D R-Tree) and is also used in order to enable effective spatiotemporal-keyword operations on semantic trajectories. There are two alternative indexing structures to support efficient STKP query processing. The proposed hybrid indices combine tightly a spatial and a text index (i.e. a 3D-Rtree and inverted file, respectively), so that both types of information can be used simultaneously for pruning the search space.

TSR-tree index. In this index, a semantic trajectory is considered as an individual unit for the tree construction. More specifically, for each semantic trajectory we compute its MBB and a list of tags related to the semantics of the episodes, sorted by time. MBB is the minimum bounding box that encloses a specific sub-trajectory of a moving object, along with the start and end times of the movement. At the end, the tags in the list are concatenated to a single string. Specifically, a pseudo-word for each semantic trajectory is created with all the concatenated tags of each trajectory's episode to a single string in order to use it for the keyword query search criteria. The leaves of the TSR-tree index are the above-described approximation of the whole semantic trajectories. For the exploitation of the graph database where our index resides, these leaf nodes are the starting nodes

of the sequence of the episodes of the approximated semantic trajectory. Moreover, inverted files (IFs) are created for all the internal nodes of the tree.

ESR-tree index. As an alternative, we build a tree using as its structural unit the episodes of the semantic trajectories rather than the semantic trajectories themselves. In other words, ESR-tree index takes into account as structural unit for the creation of the 3D-Rtree the episodes of the semantic trajectories. The leaves of the ESR-tree index are the base for the creation of the episode MBBs. Accordingly, the MBBs of the entries of the internal nodes represent the spatio-temporal union of the MBBs of the children nodes. Additionally, for each internal node there exists a pointer to an IF that organizes all the tags of its children nodes. The IFs for each internal node of the tree contain the keywords of the episodes of its child nodes.

2.2 STKP query processing

STKP queries can be processed in Hermes@Neo4j using either TSR-tree or ESR-tree index. In the former (latter) approach, STKP search algorithm takes into account that TSR-tree (ESR-tree, respectively) is built using entire semantic trajectories (episodes of semantic trajectories, respectively) as building blocks [9]. It is noteworthy that STKP search is boosted by an optimization method.

STKP query optimizer. Given a STKP query Q := <E_1, …, E_k>, where E_1, …, E_k is a sequence of spatio-temporal-textual constraints over episode abstractions, the STKP query optimizer identifies the most selective episode abstraction E^* in Q, in order to start the execution of the ESR-tree search algorithm from there, thereby pruning candidate results the earliest possible. The cost model that the query optimizer implements decomposes the computation of selectivity of an episode abstraction in two parts, one for the spatio-temporal and another for the textual component of the episode abstraction [9].

Regardless of the query length, it turns out that the search based on the ESR-tree and boosted by the query optimizer, is considerably faster, with the penalty of the higher index creation time and size compared with the TSR-tree approach.

3 SYSTEM IMPLEMENTATION

Our framework provides a robust API with the necessary tools for STD creation, querying, etc. Toward the realization of the concepts of semantic trajectories and STDs presented in the previous section, we followed the object relational (OR) approach for the datatype system of Hermes@Neo4j STD engine. In detail, we follow the abstract datatype (ADT) paradigm and define the episodes and semantic trajectory datatypes that support the definitions in [9]. Upon these datatypes, we register a rich palette of object methods; some indicative examples appear in Table 1. More details are available at Hermes@Neo4j web page[2].

Table 1: Methods over episodes and semantic trajectories

Object	Method	Definition
Episode	duration()	Returns the episode duration.
Semantic Trajectory	num_of_ episodes (String tag, String distinct), where "tag" is a set of substrings and a boolean string.	Returns the number of episodes (distinct or not, depending on distinct string) that includes tags LIKE the given ones (pattern-matching per input tag).
LayerST (object with the episodes and semantic trajectories of an STD)	confined_in (LayerST layer, MBB envelope, String tag), where tag is a concatenated set of substrings.	Returns semantic trajectories, whose episodes are overlapping spatiotemporally with the MBB and textually with the "tag" parameter.

STD creation. STD construction includes two phases (3D R-tree and IFs), along with segmental options for creating a graph database in steps (separate 3D R-tree and IFs creation routines) in case of size and memory concerns.

Semantic Trajectory Synthesizer. In case of trajectory datasets lacking textual annotations, our synthesizer is able to augment raw trajectories with textual annotations using a customized text generator that chooses terms from a lexicon of V keywords. The number of keywords for each episode follows a Zipfian distribution, in order to simulate the skewness present in real-life textual datasets.

STD search. Several functions are available for a wide range of queries like intersection, overlapping or union, with the emphasis, of course, in STKP queries (details for the appropriate search methods appear in Table 2).

Table 2: STKP query methods

Method	Parameters	Index
SpatialTemporalKeyword TrajectoryQuery	LayerST, list of MBBs,	TSR-tree index
SpatialTemporalKeyword TrajectoryEpisodesQuery	list of String tags	ESR-tree index

Hermes@Neo4j STD engine utilizes Apache Lucene [12] indexes that use inverted indexes for search and retrieval from text collections. For the implementation of interactive visualizations of the semantic trajectories over a 3D model of the globe and different types of 2D maps, the NASA WorldWind API [13] is utilized. The library has been extended in order to display the spatio-temporal and textual information of a semantic trajectory. Visual representation of search results is performed through different 3D / 2D map services, such as Open Street Map, Bing, MS Virtual Earth, NASA Blue Marble and i-cubed Landsat (Fig. 3).

The interface has the required parameters for spatio-temporal

[2] http://infolab.cs.unipi.gr/HermesNeo4j/

and textual constraints that are used as query arguments. The interface also includes necessary options for importing a dataset to a new STD. Apart from setting spatio-temporal and textual constraints, in order to perform a STKP query over the selected STD, the user decides the index and search algorithm of his/her choice. The semantic trajectories that are the results of the STKP query are displayed through a proper animation zoom at the selected map service and reference system by displaying the geographical area that covers the specific semantic trajectories. Correspondingly, information about the search results and the number of trajectories that meet the search criteria are displayed in a relevant result box.

4 ABOUT THE DEMONSTRATION

Throughout the demonstration, Hermes@Neo4j users will be able to test the system by using the real "Foursquare New York" dataset [14] and the synthetic "Hermes Attica" dataset [3] generated by the Hermoupolis generator [15]. The real dataset includes long-term (about 10 months - from Apr.12, 2012 to Feb.16, 2013) check-in data (227,428 check-ins) in New York City collected from Foursquare social network and the synthetic dataset consists of a total of 1,450,738 records that represent semantic trajectories.

Figure 3: Interactive visual exploration of a semantic trajectory that is the result of an STKP query through a 2D map representation.

The demonstration captures the following phases: (i) preparatory phase, where users have the opportunity to comprehend the internals of our framework, and (ii) our Hermes@Neo4j engine in action, where users experience various scenarios of STKP search. In particular:

Preparatory phase (background knowledge). During this phase, we show off the different datatypes and operands that can be utilized in the Hermes@Neo4j engine. In addition, we demonstrate how the user can use our framework to run legacy operands, and even more interestingly, focus on the two STKP query indexing and searching approaches.

Hermes@Neo4j engine in action. Having gained the necessary background knowledge, the user experiences a scenario of STKP search and index creation, based on the TSR-tree search algorithm. For instance, Fig. 3 present an example of an STKP query result, which is a semantic trajectory from the "Hermes Attica" dataset. The user can display the results with a selected 3D/2D map representation and reference system of his/her choice. In addition, the user can interactively switch on and off the visibility of the results. In turn, we present a scenario of STKP search and index creation, based on the ESR-tree search algorithm. The goal of this scenario is to effectively demonstrate that the STKP search based on the ESR-tree, is more efficient in comparison with the STKP search based on the TSR-tree index, with the penalty of the higher index creation time and size compared with the TSR-index.

For a deeper comprehension of the demonstrated functionality, a related video is available at Hermes@Neo4j web page[4].

ACKNOWLEDGEMENTS

This work has been partly supported by the University of Piraeus Research Center. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklowdoska-Curie agreement N. 777695.

REFERENCES

[1] Nikos Pelekis, Elias Frentzos, Nikos Giatrakos, and Yannis Theodoridis. 2008. HERMES: aggregative LBS via a trajectory DB engine. In Proceedings of SIGMOD. https://doi.org/10.1145/1376616.1376748
[2] Ralf H. Güting, Thomas Behr, and Christian Düntgen. 2010. SECONDO: a platform for moving objects database research and for publishing and integrating research implementation. IEEE Data Engineering Bulletin, 33(2):56-63.
[3] Cédric du Mouza and Philippe Rigaux. 2005. Mobility patterns. GeoInformatica, 9 (4): 297-319. https://doi.org/10.1007/s10707-005-4574-9
[4] Marcos R. Vieira, Petko Bakalov, and Vassilis J. Tsotras. 2010. Querying trajectories using flexible patterns In: Proc. of EDBT. https://doi.org/10.1145/1739041.1739091
[5] Ralf H. Güting, Fabio Valdés, and Maria L. Damiani. 2015. Symbolic Trajectories. ACM Transactions on Spatial Algorithms and Systems, 1(2). https://doi.org/10.1145/2786756
[6] Christine Parent, Stefano Spaccapietra, Chiara Renso, Gennady Andrienko, Natalia Andrienko, Vania Bogorny, Maria L. Damiani, Aris Gkoulalas-Divanis, Jose Macedo, Nikos Pelekis, Yannis Theodoridis, and Zhixian Yan. 2013. Semantic Trajectories Modeling and Analysis. ACM Computing Surveys, 45(4), article 42. https://doi.org/10.1145/2501654.2501656
[7] Nikos Pelekis, Stylianos Sideridis, and Yannis Theodoridis. 2015. Hermes^{sem}: a Semantic-aware Framework for the Management and Analysis of our LifeSteps. In: Proc. of DSAA. https://doi.org/10.1109/DSAA.2015.7344849
[8] Stylianos Sideridis, Nikos Pelekis, and Yannis Theodoridis. 2016. On querying and mining semantic-aware mobility timelines. International Journal of Data Science and Analytics, 2(1-2), 29-44 (2016). https://doi.org/10.1007/s41060-016-0030-1
[9] Fragkiskos Gryllakis, Nikos Pelekis, Christos Doulkeridis, Stylianos Sideridis, and Yannis Theodoridis. 2017. Searching for Spatio-Temporal-Keyword Patterns in Semantic Trajectories. In: Proc. of IDA. https://doi.org/10.1007/978-3-319-68765-0_10
[10] Neo4j graph database, https://neo4j.com/.
[11] Neo4j Spatial library, http://neo4j-contrib.github.io/spatial/.
[12] Apache Lucene, https://lucene.apache.org/.
[13] NASA WorldWind, https://worldwind.arc.nasa.gov/.
[14] Dingqi Yang, Daqing Zhang, Vincent W. Zheng, and Zhiyong Yu. 2015. Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs. TSMC, 45(1). https://doi.org/10.1109/TSMC.2014.2327053
[15] Nikos Pelekis, Stylianos Sideridis, Panagiotis Tampakis, and Yannis Theodoridis. Simulating our LifeSteps by Example (2016) ACM Trans. Spatial Algorithms and Sytems, 2(3), article 11. https://doi.org/10.1145/2937753

[3] http://chorochronos.datastories.org/?q=content/hermes-attica

[4] http://infolab.cs.unipi.gr/HermesNeo4j/visual.htm