DLEEL: Multi-Predicate Spatial Queries on User-generated Streaming Data

Abdulaziz Almaslukh*, Laila Abdelhafeez‡, Amr Magdy§
Department of Computer Science and Engineering & Center for Geospatial Sciences
University of California, Riverside
Email: *aalma021@ucr.edu, ‡labde005@ucr.edu, §amr@cs.ucr.edu

***Abstract*—This paper demonstrates *DLEEL*; a research system that supports scalable spatial queries with multiple predicates on user-generated data streams, such as social media streams. Supported queries include spatial-social queries and spatial-keyword queries, which are popular in different applications but have never been addressed in the challenging environment of streaming data, where data arrives with excessively high rates. *DLEEL* distinguishes itself with three novel contributions: (1) Indexing spatial-social data in for personalized real-time search: *DLEEL* is the first to address personalized queries on streaming spatial-social data through novel low-overhead indexing that scales for large amounts of data and users. The novel indexing has a hybrid storage architecture that trades off indexing overhead, memory consumption, and query latency. (2) Indexing spatial-keyword data for real-time search: *DLEEL* is the first to enrich existing spatial-keyword indexes with novel streaming data components. The new components reveal performance losses and gains from a system perspective, trading off the system overhead with flexibility to support a variety of queries. (3) Scalable query processing: *DLEEL* exploits the indexes content to smartly prune the search space on multiple dimensions and support efficient query latency for its different queries on excessive number of data records. *DLEEL* is demonstrated using a stream of 5 billions real tweets collected from Twitter APIs and real query locations obtained from a popular web search engine. *DLEEL* has shown superior performance with serving incoming queries with an average latency of few milliseconds while digesting hundreds of thousands of data records every second.**

I. Introduction

User-generated streaming data are generated from popular web-based platforms such as Twitter, Facebook, and Foursquare. This data has drawn significant attention in the last few years due to introducing new research challenges from a data management perspective [7]. Spatial queries have got particular attention [3], [6] due to the availability of location information in this data, where 80+% of users post from mobile devices. Keyword and social attributes support major applications on this data. So, spatial-keyword queries and spatial-social queries, e.g., *"find what my friends post in Paris"* or *"find what users post in The Bahamas about help and rescue"*, have several applications in rescue, disaster management, finding real-time local news, and real-time personalized recommendations [1], [2]. As a result to their importance, some of these queries have recently started to make it to the system-level support [4] to allow application developers to build on top of relatively stable datasets. However, there is no research focus on supporting streaming data, which is our main focus.

This paper demonstrates *DLEEL*; a system that supports a variety of scalable social spatio-temporal and textual queries on user-generated streaming data. *DLEEL* indexes excessive numbers of incoming data in real time and supports queries that satisfy multiple predicates including spatial, temporal, textual, and user's social network predicates. The supported queries are top-k extensions to the two fundamental spatial queries, range query and k-nearest-neighbor (kNN) query, to retrieve the most k useful records according to a certain ranking function. The main contributions of *DLEEL* can be summarized as follows:

(1) **Indexing for spatial-social queries on streaming data**: *DLEEL* is the first to introduce a novel indexing paradigm on streaming data for personalized queries based on both spatial and social dimensions to serve spatial-social queries at scale [2]. Thus, *DLEEL* gets personalized answers from the query issuer's social network with minimal system overhead to handle hundreds of millions of users with affordable resources and latency. Personalized queries could also include optional keywords to *"find what my friends post about demonstrations in Paris"*, for example. The indexing has a hybrid storage architecture where both main-memory and disk storage are combined to trade off indexing overhead, memory consumption, and query latency. In addition, the social information is incorporated as a discrete social distance that personalizes the answer and still provides light data management overhead. The details of our techniques are presented in [2].

(2) **Indexing for spatial-keyword queries on streaming data**: *DLEEL* is the first to enrich existing spatial-keyword indexes [1], [5] with streaming components. *DLEEL* indexing framework reveals performance implications of different design decisions from a system perspective, trading off the indexing overhead with flexibility to support a variety of queries. This gives insights for system builders and administrators on efficient processing and optimization of these queries in terms of data digestion rate, memory consumption, and query latency. The details of out techniques are presented in [1].

(3) **Scalable query processing**: On top of the indexing components, *DLEEL* provides scalable query processing techniques that efficiently prune the search space to provide low query latency on excessive number of data records in real time. All supported queries include the temporal aspect due to the real-time nature of streaming data. So, *DLEEL* prunes both

2375-026X/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDE48307.2020.00166

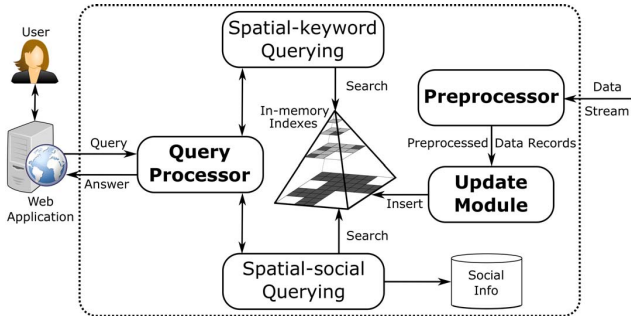

Fig. 1. *DLEEL* Architecture.

spatial and temporal dimensions, in addition to other query dimensions, either textual or social dimensions. Section IV summarizes our techniques that are detailed in [1], [2].

DLEEL is demonstrated with an actual system prototype using a stream of real tweets simulated from a large archive of 5 billion tweets collected during 2014-2019, with an actual query workload obtained from a popular web search engine.

II. DLEEL Overview

System architecture. Figure 1 shows *DLEEL* system architecture. *DLEEL* has three main modules, *preprocessor*, *update module*, and *query processor*, centered around *in-memory index structures* that serve both spatial-keyword and spatial-social queries. The incoming streaming data is preprocessed to extract the needed information such as geo-location, keywords, timestamp, and posting user. Then, the preprocessed data is forwarded to the update module that is equipped with efficient insertion techniques to continuously insert new data into the in-memory indexes with fast rates. Meanwhile, users are posting queries to *DLEEL* through a web application that submits the queries to the query processor module. The query processor has two prominent sub-components that efficiently process spatial-keyword and spatial-social queries, respectively. The incoming query is directed to the appropriate querying module to get the query results that are routed back to end users through the web application.

Supported queries. *DLEEL* supports four queries:

(1) *Spatial-social Temporal Range Query (SSTRQ)*: given <spatial range R, keywords W, integer k, user u>, SSTRQ retrieves the most recent k records within R that contain W and posted by u's friends or friends-of-friends.

(2) *Spatial-social Temporal kNN Query (SSTkQ)*: given <spatial location L, keywords W, integer k, timestamp T, user u>, SSTkQ retrieves top-k records that contain W, posted by u's friends or friends-of-friends, and ranked based on a spatio-temporal distance from L and T.

(3) *Spatial-keyword Temporal Range Query (SKTRQ)*: given <spatial range R, keywords W, integer k>, SKTRQ retrieves the most recent k records within R that contain W.

(4) *Spatial-keyword Temporal kNN Query (SKTkQ)*: given <spatial location L, keywords W, integer k, timestamp T >, SKTkQ retrieves top-k records that contain W and ranked based on a spatio-temporal distance from L and T.

All queries are top-k and temporal due to the nature of streaming data that mandates returning only k useful records, ranked based on a certain ranking function, rather than an overwhelming number of records, as detailed in [1], [2].

III. Indexing

DLEEL supports two indexes: spatial-social and spatial-keyword indexes. All data indexes are wholly residents in main-memory for scalable data digestion and high-throughput queries. Only the social component of the spatial-social index, that maintains relatively stable information, is divided between memory and disk. The two indexes are briefly outlined below.

Spatial-social index. *DLEEL* proposes a novel spatial-social index for streaming data and evaluates it against existing geo-social indexes in the literature that mostly work for stable datasets. The index distinguishes highly-dynamic data, that is represented by streaming data records, from relatively stable data, that is represented by the social graph of users that changes infrequently. The streaming data records are continuously digested in a spatial-user index that is wholly resident in main-memory. The in-memory index is based on a spatial quad-tree, where each quad-tree node holds a hash index that organizes data based on the posting user. The social information is wholly resident in disk while an in-memory buffer is used to swap-in social information that are used in incoming queries. The social graph is stored as a set of light friend lists that is used by the query processor to support efficient computation of a discrete social distance. Our techniques are detailed in [2].

Spatial-keyword index. *DLEEL* extends the exiting spatial-keyword indexes for streaming data with light batch insertion techniques so that hundreds of thousands of data records can be digested efficiently every second. *DLEEL* applies this to ten different spatial-keyword indexes that are composed of different combinations of four building blocks: spatial grid, spatial quad-tree, spatial R-tree, and inverted index. Pure and hybrid index structures are composed using these four blocks and evaluated on excessive streaming data to reveal the relative performance gains of each from a system perspective in terms of indexing scalability, main-memory consumption, and query latency. This provides system builders with insights on the abilities of different blocks to handle streaming data at scale at a system-level. The study is detailed in [1].

IV. Query Processing

To process the supported queries in *DLEEL* (as defined in Section II), the query processor takes advantage of the underlying index structures to efficiently prune the search space on both spatial and temporal dimensions, in addition to other query attributes such as keyword and social dimensions. The spatial index is first used to get a set of spatial tiles within the query range or near the query location. Initial k records are retrieved, then new pruning ranges are computed to eliminate any record that cannot make it to the final answer. As the result set is refined by adding new data records, the

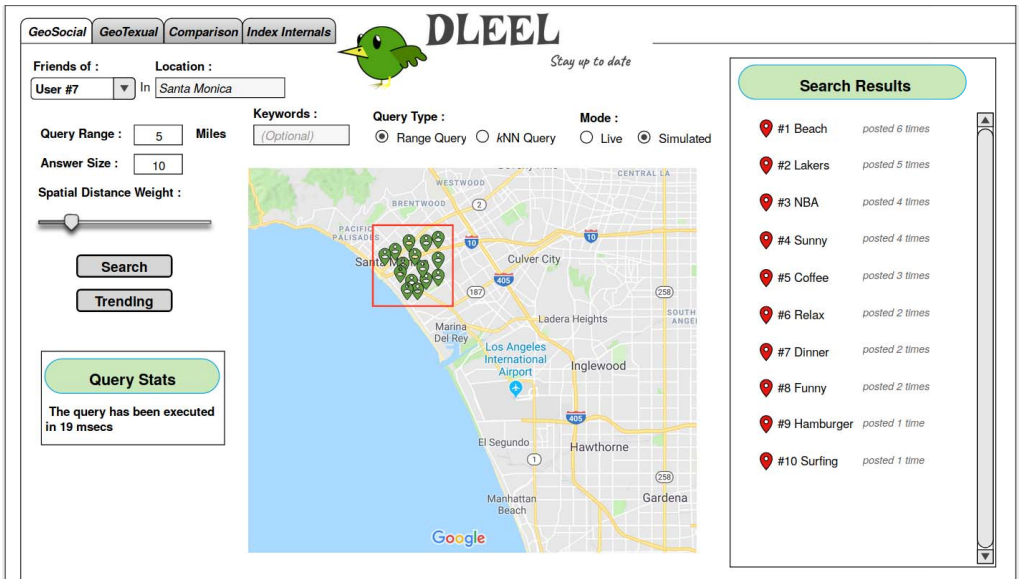

Fig. 2. Spatial-social Trending Query.

pruning ranges are tightened and more records are pruned. The underlying indexes organize data based on temporal, keyword, and social dimensions, which allows all dimensions to be used simultaneously to prune the search space in many cases. Such powerful pruning allows *DLEEL* query processor to provide low query latency, within milliseconds, for different queries with significant improvements to existing competitors [1], [2].

V. Demonstration Scenarios

DLEEL functionality and internals are demonstrated with real tweets as a prime example for user-generated streaming data. All demonstration scenarios can operate in either **live mode** or **simulation mode**. The *live mode* brings online geo-tagged tweets *from the demo attendee's network of friends* in real time through Twitter APIs, which provide small number for public users. To show *DLEEL* scalability, the *simulation mode* uses a real dataset of 5 billions geo-tagged tweets, collected over the past five years from the same Twitter APIs to simulate a data stream with high arrival rates (85K tweets/sec). Our demo attendees would be able to interact with *DLEEL* through one or more of the following scenarios.

A. Scenario 1: Spatial-Social Queries

Our demo attendees can issue spatial-social queries to get tweets posted by her friends (or other user's friends) in certain locations, and optionally related to certain keywords. Figure 2 depicts *DLEEL* web interface to post these queries. The interface allows posting either range or kNN social queries, that are defined in Section II. The issuing user is either predefined and selected from a drop down list or the demo attendee could input her Twitter username with authentication, then she inputs a geographic location, which can be directly specified on the map or navigated through the location text box. The attendee can also add optional keywords to its corresponding text box. Moreover, she can optionally change the default parameters values of the answer size k, spatial range, and the spatial distance weight α, $0 \leq \alpha \leq 1$, by changing the query parameter boxes and a slider. The spatial range is used as the primary spatial parameter for the range queries, while in kNN queries it is used as the maximum range of spatial distance from where answer tweets can be retrieved. Once the user clicks the search button, *DLEEL* backend will process the query, through the spatial-social querying module, and return the results to the front-end interface to be displayed on the map view and a side textual panel. The interface also shows the query latency measured from the backend modules.

B. Scenario 2: Spatial-Keyword Queries

DLEEL demo attendees can also post spatial-keyword queries to get tweets that are related to certain keywords in certain locations without the involvement of their social network. Figure 3 depicts the user interface of *DLEEL* spatial-keyword queries. The user can select to issue either a range or a kNN query. Then, she inputs one or more keywords that tweets should have in addition to a geographic location, either through the map or through the location text box. She can also change the default value of the spatial range, answer size k, and the spatial distance weight, similar to Scenario 1. Finally, the user clicks the search button to issue the query that is processed in the back-end spatial-keyword querying module and the results are displayed by the front-end in a right-sided results panel as shown in Figure 3.

C. Scenario 3: Trending Queries

On top of both spatial-social and spatial-keyword queries, our demo attendees can issue analytical queries which summarize what is trending in the region by finding the most frequent keywords, either within the user's friends or based on general tweets with certain keywords. For spatial-social queries, this query helps users to find out what friends are

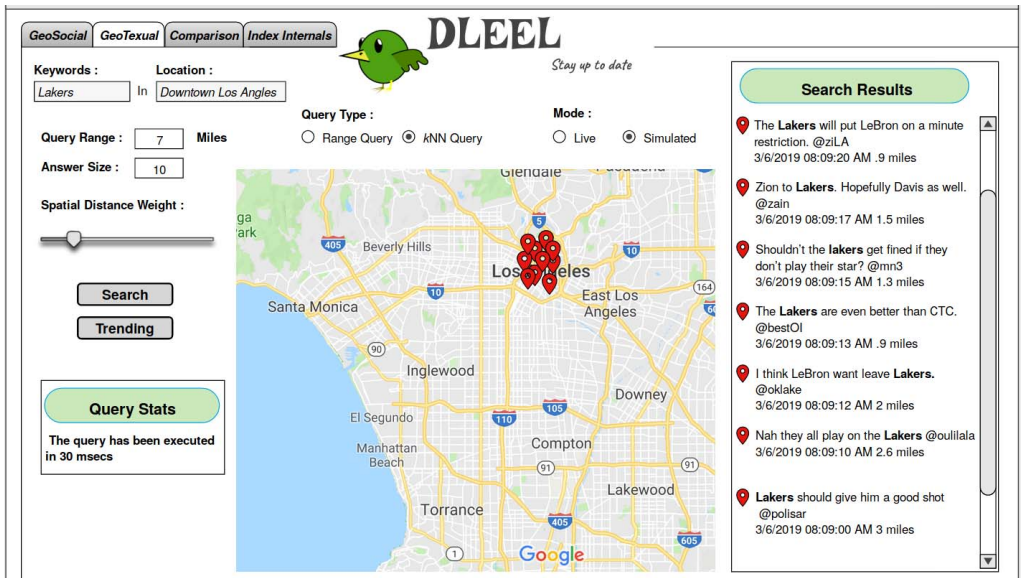

Fig. 3. Spatial-keyword kNN Query.

talking about, and for spatial-keyword queries, it helps figuring out on-going related topics, e.g., everyone who mentions *Trump* today mentions *Turkey's Erdogan* as well. For this query, *DLEEL* users will use the web interface that is used in the previous two scenarios, yet, will click on *trending* button instead of *search* button. To process the query, *DLEEL* back-end processes either a spatial-social or a spatial-keyword query with the input parameters, then the results are sent to an analytical module that performs additional processing to reveal the trending keywords. The trending keywords are displayed on the side panel and locations of users (if any) are displayed on the map view as shown in Figure 2.

D. Scenario 4: Comparing Multiple Indexes

Our demo shows the difference of employing one index over the others in terms of digestion rate, memory consumption, and query latency. A dedicated user interface is built to allow the demo attendees try different indexes for both spatial-keyword and spatial-social queries. The user chooses multiple of the available indexes and the query type, then the user clicks *compare* button to start indexing and querying a sample of the tweets. All measurements information are displayed to the user giving insights about relative performance gains and losses of different indexes from a system point of view.

E. Scenario 5: Index Internals

DLEEL demonstrates the indexing internals through showing the batch update operation with an animated map object that is depicted in Figure 4. *DLEEL* supports bulk insertion which aggregate a batch of new tweets that are ready to be inserted in the index. This batch will be divided according to the minimum bounding rectangle of the cells starting from the root to the leaves. Different variations of this operation are performed for different index structures, and it improves

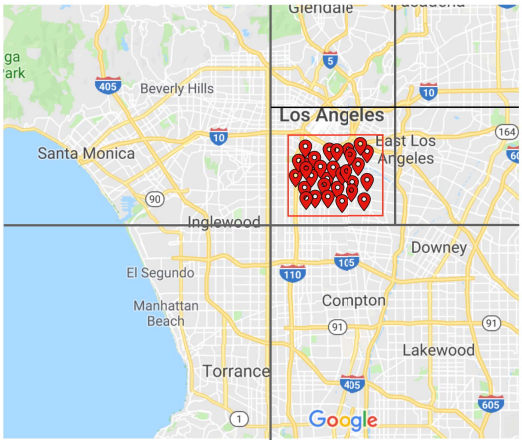

Fig. 4. *DLEEL* Batch Insertion.

the digestion rate of streaming data significantly in real time. Figure 4 shows part of quad-tree nodes with a batch of data inserted into Los Angeles greater area. Our demo shows the live update of the index when inserting a batch of new tweets until the batch is completely digested.

References

[1] A. Almaslukh and A. Magdy. Evaluating Spatial-Keyword Queries on Streaming Data. In *ACM SIGSPATIAL*, 2018.

[2] A. Almaslukh and A. Magdy. Temporal Geo-Social Personalized Search Over Streaming Data. In *ACM SIGSPATIAL*, 2019.

[3] C. Budak et al. GeoScope: Online Detection of Geo-Correlated Information Trends in Social Networks. In *VLDB*, 2014.

[4] M. J. Carey. AsterixDB Mid-Flight: A Case Study in Building Systems in Academia. In *ICDE*, 2019.

[5] L. Chen, G. Cong, C. S. Jensen, and D. Wu. Spatial Keyword Query Processing: an Experimental Evaluation. In *VLDB*, 2013.

[6] W. Feng et al. STREAMCUBE: Hierarchical Spatio-temporal Hashtag Clustering for Event Exploration Over the Twitter Stream. In *ICDE*, 2015.

[7] A. Magdy, L. Abdelhafeez, Y. Kang, E. Ong, and M. F. Mokbel. Microblogs Data Management: A Survey. *VLDB Journal*, 2019.