Location- and keyword-based querying of geo-textual data: a survey

Chen, Zhida; Chen, Lisi; Cong, Gao; Jensen, Christian S.

Published in:
VLDB Journal

DOI (link to publication from Publisher):
10.1007/s00778-021-00661-w

Publication date:
2021

Document Version
Accepted author manuscript, peer reviewed version

Citation for published version (APA):
Chen, Z., Chen, L., Cong, G., & Jensen, C. S. (2021). Location- and keyword-based querying of geo-textual data: a survey. *VLDB Journal*, *30*(4), 603-640. https://doi.org/10.1007/s00778-021-00661-w

Noname manuscript No.
(will be inserted by the editor)

Location and Keyword Based Querying of Geo-Textual Data: A Survey

Zhida Chen · Lisi Chen · Gao Cong · Christian S. Jensen

Received: date / Accepted: date

Abstract With the broad adoption of mobile devices, notably smartphones, keyword-based search for content has seen increasing use by mobile users, who are often interested in content related to their geographical location. We have also witnessed a proliferation of geo-textual content that encompasses both textual and geographical information. Examples include geo-tagged microblog posts, yellow pages, and web pages related to entities with physical locations. Over the past decade, substantial research has been conducted on integrating location into keyword-based querying of geo-textual content in settings where the underlying data is assumed to be either relatively static or is assumed to stream into a system that maintains a set of continuous queries. This paper offers a survey of both the research problems studied and the solutions proposed in these two settings. As such, it aims to offer the reader a first understanding of key concepts and techniques, and it serves as an "index" for researchers who are interested in exploring the concepts and techniques underlying proposed solutions to the querying of geo-textual data.

Z. Chen
School of Computer Science and Engineering
Nanyang Technological University
Singapore
E-mail: chen0936@e.ntu.edu.sg

L. Chen
Inception Institute of Artificial Intelligence
Abu Dhabi, United Arab Emirates
E-mail: lisi.chen@inceptioniai.org

G. Cong (✉)
School of Computer Science and Engineering
Nanyang Technological University
Singapore
E-mail: gaocong@ntu.edu.sg

C. S. Jensen
Department of Computer Science
Aalborg University
Denmark
E-mail: csj@cs.aau.dk

Keywords Geo-textual data · Spatio-textual data · Spatial keyword query · Survey

1 Introduction

We have witnessed a rapid proliferation of geo-textual, or spatio-textual, data over the last decade. One example of such data is web pages with associated geographical information. Other examples include geo-tagged microblog posts (e.g., geo-tagged tweets from Twitter[1]) that contain both text and location information; geo-tagged photos from social photo sharing services (e.g., Flickr[2] and Instagram[3]) that host photos with both descriptive tags and geographical information; check-ins from location-based social networks (e.g., Foursquare[4]); reviews of Points of Interest (POI) on local-business websites (e.g., Yelp[5] and TripAdvisor[6]) containing both text and locations; and online local or regional news comprising text documents and location tags.

Such geo-textual content is available from a range of sources [46]. By 2014, more than 40 million georeferenced photos were posted on Flickr, and over one million geo-tagged articles were available on Wikipedia [114]. More than 10 million geo-tagged

[1] https://twitter.com/
[2] https://www.flickr.com/
[3] https://www.instagram.com/
[4] https://foursquare.com/
[5] https://www.yelp.com/
[6] https://www.tripadvisor.com/

tweets are posted daily on Twitter [160], and some 8 million check-ins are submitted to Foursquare per day [1]. By June, 2019, the cumulative number of reviews of businesses on Yelp reached 192 million [6]. In addition to the textual and geo-spatial information, geo-textual content may include a wide range of other information, such as the user who posted the content, the content creation time, content category information, and ratings.

Next, with the proliferation of smartphones, we have witnessed an increasing demand for easy access to geo-textual data. Specifically, the querying of geo-textual data has occurred in two settings: one where the data is assumed to be relatively static, but where queries that need to be processed arrive continuously; and one where the data streams into the system that needs to maintain up-to-date answers to a relatively static or dynamic set of continuous queries.

In the first setting, updates, including insertions and deletions, to the data occur at a relatively low pace. A key challenge is to be able to process high volumes of incoming queries with low latency. While the queries in the first setting are conventional one-time queries, the queries in the second setting are continuous queries—queries that are registered in the system and for which up-to-date results need to be maintained as data streams into the system until the queries are deregistered. Such queries are also called standing queries.

Many studies in both settings are reported in the literature. This paper covers the specific problem definitions in these settings and also offers insight into the solutions provided, thus offering an in-depth survey of studies in the two settings.

Compared with previous surveys related to spatial keyword queries [20, 36, 47, 109, 111], this study is more comprehensive in that it covers a broader range of queries and presents the high-level ideas underlying the algorithms proposed for answering these queries. Previous surveys cover fewer types of queries and do not cover the ideas of the proposed algorithms. For example, Cao et al [20] do not cover querying in road networks (Section 3.2) and querying streaming data (Section 5), Mahmood et al [111] discuss geo-textual indices that were proposed between 2010 and 2017, but do not cover query definitions or algorithms, Chen et al [36] do not cover many standard query extensions (Section 3.1.2) and localized event detection(Section 5.2), and the book [109] does not cover many standard query extensions (Section 3.1.2) and query modification (Section 3.1.6).

A notable recent book [109] offers an overview of existing centralized and distributed solutions to spatial keyword querying. It classifies existing studies according to the spatial keyword predicates used in the queries, including spatial keyword selection predicates, spatial keyword group predicates, spatial keyword join predicates, continuous spatial keyword queries, and aggregate spatial keyword predicates. We offer a more complete coverage of existing work and also aim to capture the relations and differences among existing queries. The book puts emphasis on specific studies and features them in case studies. In contrast, we offer a broader high-level overview of proposed algorithms for answering the different types of queries. We classify studies based on the settings, i.e., whether a standard database setting or a streaming setting is assumed. For each setting, we separate the query definitions from the proposed algorithms for answering the queries. This enables a more complete and in-depth coverage within the chosen scope, which brings out the relations and differences among different kinds of queries.

A recent survey by Chen et al. [36] covers spatial keyword search over geo-textual data. It categorizes existing studies as querying individual geo-textual data and querying connected geo-textual data. Compared with our survey, it covers much fewer types of queries. Furthermore, it covers the high-level ideas of algorithms for trajectory search, not for the types of queries that are the focus of our survey. Our survey thus covers a broader range of queries and the high-level ideas of the related algorithms, and it also describes the relations among the queries covered. In contrast, the survey [36] is limited to query definitions. In addition, this survey focuses on geo-textual data with point or region locations, but does not cover geo-textual trajectory data [36, 65].

To summarize, the survey is designed to offer a compact yet rich coverage of important concepts and techniques related to the location and keyword based querying of geo-textual data. It is comprehensive in its coverage of different types of queries and brings out the key concepts underlying the proposed solutions. As such, the survey serves as an "index" for researchers who are interested in exploring the concepts and techniques underlying proposed solutions to the querying of geo-textual data.

The reminder of the survey is organized as follows: Section 2 covers the types of data considered as well as the categorization framework that we adopt. Section 3 presents problem definitions of existing studies on the querying in the conventional setting of static geo-textual data. We first cover problems that assume an underlying Euclidean space and then cover problems that assume a spatial network. For each of these spaces, we apply the categorization framework. We first consider standard spatial keyword queries in Euclidean space and then consider other types of Euclidean-space queries. We present connections between these and the standard queries whenever possible. We end by covering the studies that assume a spatial network. Section 4 proceeds to cover concepts and techniques underlying the solutions to the problems covered in Section 3. Section 5 covers problem definitions in the setting of streaming geo-textual data. Here, we again structure the definitions according to the classification framework. Section 6 presents the concepts and techniques underlying the solutions to the problems covered in Section 5. Section 7 offers conclusions.

2 Data and Classification Framework

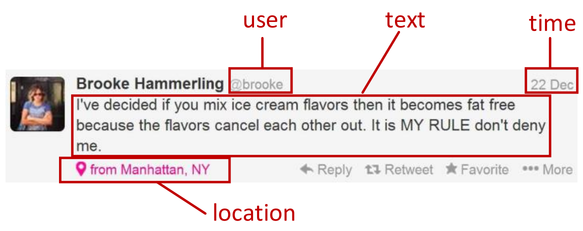
2.1 Geo-textual Data

The geo-textual data considered in the survey comprise a range of geo-textual objects. A geo-textual object has a location and a textual attribute and may have additional attributes.

Location. The location attribute of a geo-textual object can take two forms: (1) a *physical location*, which specifies a location by means of GPS coordinates (e.g., 40°15′52″ N, 112°34′16″ W); and (2) a *semantic location*, which can be a sequence of hierarchically organized geo-keywords (e.g., 57E Healey St, Champion, IL, U.S.A.) or a POI name (e.g., Eiffel Tower). Locations are typically modeled as point locations. However, locations may also have an extent. Specifically, physical locations are generally modeled as points, and semantic locations are modeled either as points or regions.

Text content. The text attribute of a geo-textual object is generally represented by a term vector or a bag of terms.

Time attribute. In some cases, geo-textual objects have a time attribute. The time attribute is often a time point (e.g., 10:05:20 a.m., 30-MAY-2019), capturing the time when the real-world object represented by the geo-textual object was at the indicated location or the time when the object was created. The literature uses terms such as *spatio-temporal document* or *spatio-temporal message* for such objects. Common examples include geo-tagged tweets from Twitter (Fig. 1), check-ins at particular Points of Interest, and location reports by individuals who move about. Time attributes may also include time periods (e.g., opening hours: 10:00 a.m. to 9:00 p.m., weekdays).



Brooke Hammerling @brooke 22 Dec
I've decided if you mix ice cream flavors then it becomes fat free because the flavors cancel each other out. It is MY RULE don't deny me.
from Manhattan, NY

Fig. 1 Geo-tagged Tweet

2.2 Classification Framework

Fig. 2 provides an overview of the classification scheme adopted in the survey. Existing studies relate to one of two settings.

Querying Static Geo-textual Data. Studies in this setting concern the search and exploration of a collection of geo-textual objects that are relatively static and are updated infrequently. In this setting, we classify existing studies based on how the underlying space is modeled: as an Euclidean space or a road network. The main difference between the two types of studies is how the distance between two spatial points is computed: as Euclidean distance or network distance. We further classify the Euclidean space studies into six categories: standard queries, standard query extensions, group queries, region finding/analysis queries, spatio-textual join queries, and query modification. We also classify the road network space studies into five categories: standard queries, standard query extensions, socially-aware queries, group queries, and route planning queries.

Example: Retrieve all objects whose text contains *food* and whose location is within 3 km of the *Hyatt Regency Hotel, San Francisco, U.S.A.*.

Querying Streaming Geo-Textual Data. In this setting, the data arrives at the system "continuously," meaning at high frequency or speed. Studies in this setting are divided into four categories: location-based publish/subscribe functionality, localized event detection, temporal spatial keyword queries, and location-based term queries.

Example: A user submits a subscription query to a publish/subscribe system to get notified of every new tweet mentioning *iphone* and whose location is within 3 km of the user's home.

As already indicated, studies on the querying of geo-textual data model geographical space as either Euclidean space or network space.

When a Euclidean space is adopted, the space is typically two-dimensional, and object locations are typically modeled as point locations. Euclidean distance or

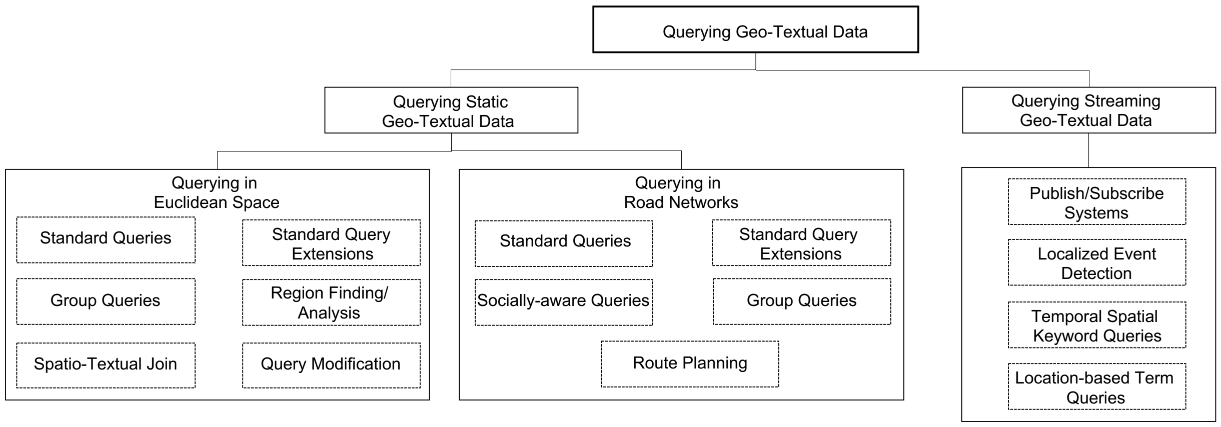

Querying Geo-Textual Data
Querying Static Geo-Textual Data
Querying Streaming Geo-Textual Data
Querying in Euclidean Space
Standard Queries
Standard Query Extensions
Group Queries
Region Finding/ Analysis
Spatio-Textual Join
Query Modification
Querying in Road Networks
Standard Queries
Standard Query Extensions
Socially-aware Queries
Group Queries
Route Planning
Publish/Subscribe Systems
Localized Event Detection
Temporal Spatial Keyword Queries
Location-based Term Queries

Fig. 2 Geo-Textual Query Classification Framework

Squared Euclidean distance, denoted by $\|\cdot\cdot\|$ and $\|\cdot\cdot\|^2$, respectively, are used as the distance notion.

When a spatial network is adopted, the motivation is to capture that object or user movement occurs in a road network and that objects are reachable via a road network. More specifically, a road network is modeled as a spatial network. Spatial network models come in a number of different variations. The most common one may be that of a regular undirected or directed graph where each vertex has a Euclidean point location and where each edge has a weight that captures the edge's length. The position of a geo-textual object is often given by an edge and a distance from the start of the edge, or it is simply assumed that objects can only be located at vertices, in which case a position is given by a vertex. In spatial network models, the relevant distance notion is spatial network distance. The distance from a source object to a target object is then the length of the shortest path from the source object to the target object. In the context of spatial networks, travel time may also be considered as the relevant distance notion. In that case, travel times, sometimes time varying, are associated with the edges.

3 Problem Definition for Querying Static Geo-textual Data

3.1 Querying in Euclidean Space

We proceed to cover the problems addressed in Euclidean space according to the six categories presented in Fig. 2. The standard queries are the most common and include Boolean range, Boolean kNN, and top-k kNN spatial keyword queries. The standard query extensions consider additional aspects, such as temporal aspects and social relations. The group queries consider inter-object relations and return sets of objects that collectively satisfy a query. The region finding/analysis queries find sets of regions satisfying pre-defined constraints, or they investigate the data distribution in a region. The work on query modification relates to the refinement of spatial keyword queries to improve user satisfaction with the results.

3.1.1 Standard Queries

In general, a standard spatial keyword query takes as input a spatial parameter (s) and a textual parameter (t), and it returns one or more geo-textual objects, which can be either rank-ordered or not.

Spatial parameter. The typical spatial parameter is a point location that models the location of the user who issues the query, or a location of interest. The spatial parameter can also be a set of point locations, or a region.

Textual parameter. The textual parameter takes two different forms: a Boolean keyword expression or a set of keywords. A Boolean keyword expression consists of a set of keywords connected by AND or OR operators. It is used for finding objects whose textual content satisfies the expression. The textual parameter in the form of a set of keywords is adopted mostly in top-k queries that rank the objects based on a function that quantifies the textual relevance between the query and the objects.

The standard spatial keyword queries either involve only Boolean predicates on the spatial and textual (and possibly other) object attributes, or they return the top-k objects that satisfy optional Boolean predicates according to a ranking function. Specifically, first, a spatial keyword Boolean predicate takes the form $P_{st}(s, t, o)$, where s and t are spatial and textual parameters, and o is a geo-textual object. Such a predicate can be expressed as a conjunction of a spatial predicate and a textual predicate: $P_s(s, o)$ and $P_t(t, o)$. Range and containment predicates are common spatial and textual predicates, respectively. The cardinality of the result may vary from 0 to the size of the set of objects. Second, for a ranking query, the ranking function can be any function that takes an object as an argument and assigns a score to it.

Let D be a set of geo-textual objects. Each object $o \in D$ is defined as a pair $(o.\rho, o.\psi)$, where $o.\rho$ is a 2-dimensional Euclidean point location and $o.\psi$ is its text content. We consider different types of queries based on the ways they use spatial and textual predicates.

Boolean range spatial keyword (BRSK) query [41, 45, 49, 68, 83, 102, 135, 163, 177]. A BRSK query $q = (BE, \rho, r)$ takes three parameters: BE is a Boolean keyword expression that is composed of a set of keywords connected by AND or OR operators, ρ is a query location, and r is a query region radius. A BRSK query can be considered as the combination of a Boolean query from information retrieval and a range query from spatial databases. Formally, the result $q(D)$ of q is the subset of D satisfying

$$\forall o \in q(D)(dist(o, q) \leq q.r \wedge q.BE(o.\psi)).$$

Here, $q.BE$ can be represented by $q.\psi_1 \vee q.\psi_2 \vee \cdots \vee q.\psi_m$, where $q.\psi_i$ $(1 \leq i \leq m)$ is a set of query keywords, and $q.BE(o.\psi)$ returns true if $\exists 1 \leq i \leq m(q.\psi_i \subseteq o.\psi)$.

Most existing studies (e.g., [41, 45, 49, 68, 102, 135, 177]) consider a simplified BRSK query that considers a Boolean keyword expression composed of AND operators only. This query is given by $q' = (\psi, \rho, r)$, and the result $q'(D)$ of q' is the subset of D satisfying

$$\forall o \in q'(D)(dist(o, q) \leq q.r \wedge q.\psi \subseteq o.\psi).$$

Example: Retrieve all objects whose text contains keywords *vegeterian*, *pizza*, and *quiet* and whose location is within 3 km of the *Hyatt Regency Hotel, San Francisco, U.S.A.*

One study considers an approximate keyword range query [16], which has an approximate keyword constraint (e.g., based on edit distance or Jaccard similarity). This query finds objects that are textually more similar to the query keywords than a threshold and belong to a given spatial query range.

A BRSK query may retrieve any number of objects, and the result objects are not ranked. However, a BRSK query can be modified easily to rank the result objects based on a scoring function, and it is easy to extend query processing algorithms to achieve this. The extension returns up to k objects that are located in the query region, ranked according to a ranking function. The ranking function can be a linear combination of spatial proximity and textual relevance. We call such a query a top-k range spatial keyword (kRSK) query [41, 103, 163, 177]. This query can be considered as a combination of a ranking query from information retrieval and a range query from spatial databases.

Boolean kNN spatial keyword (BkSK) query [25, 76, 98, 133, 146, 158, 163]. A BkSK query $q = (BE, \rho, k)$ takes three parameters, where BE is a Boolean keyword expression as stated in the BRSK query, ρ is a spatial point, and k is the number of objects to retrieve. The query combines a Boolean keyword query from information retrieval and a kNN query from spatial databases. The result $q(D)$ of a BkSK query is a set of (at most) k objects, each of which satisfies the Boolean keyword expression $q.BE$. The objects are ranked according to their distances to ρ. Formally, $\forall o \in q(D)((\nexists o' \in D \setminus q(D))(dist(o', q) < dist(o, q)) \wedge q.BE(o.\psi))$.

Example: Retrieve the k objects nearest to the *Hyatt Regency Hotel, San Francisco, U.S.A.* such that each object's text contains the keywords *sushi* and *ramen*.

One study [163] employs a ranking function that is a linear combination of the spatial proximity and textual relevance between q and o.

Top-k kNN spatial keyword (TkSK) query [48, 59, 91, 122, 152, 153, 158, 163]. A TkSK query $q = (\psi, \rho, k)$ takes three parameters: ψ is a set of keywords, ρ is a spatial point, and k is the number of objects to retrieve. The query result $q(D)$ is a set of (at most) k objects. The objects are ranked according to a score that takes into consideration spatial proximity and textual relevance. Formally, $\forall o \in q(D)((\nexists o' \in D \setminus q(D))(ST(o', q) < ST(o, q)))$, where the ranking score $ST(o, q)$ can be defined by $ST_a(o, q)$ [48, 59, 91, 152, 158, 163] or $ST_b(o, q)$ [122] that are defined as follows:

$$ST_a(o, q) = \alpha \cdot ss(o, q) + (1 - \alpha) \cdot (1 - st(o, q)) \tag{1}$$

$$ST_b(o, q) = \frac{ss(o, q)}{st(o, q)} \tag{2}$$

Here, $ss(o, q)$ is the spatial proximity of o to q, $st(o, q)$ is the textual relevance of o to q, and $\alpha \in [0, 1)$ in Eq. 1 is a preference parameter that makes it possible to balance spatial proximity and textual relevance. In Eq. 2, the combination between spatial proximity and textual relevance is represented as a fraction, thus eliminating the query preference parameter.

In these definitions, the spatial proximity is defined as the normalized Euclidean distance: $ss(o, q) =$

$\frac{dist(o,q)}{dist_{max}}$, where $dist(\cdot,\cdot)$ denotes Euclidean distance, and $dist_{max}$ is the maximum distance between any two objects in D. Further, the textual relevance $st(\cdot,\cdot)$ can be computed using an information retrieval model, such as the language model (e.g., [48]), cosine similarity (e.g., [122]), or BM25 (e.g., [45]) that is normalized to a scale similar to the spatial proximity.

One study [81] considers a different TkSK query where the spatial part of an object is a region (i.e., a rectangle or another shape) and where the spatial part of the query is a set of rectangles. The ranking score is computed using Eq. 1.

Example: Retrieve the k objects with the highest ranking scores with respect to the location of the *Hyatt Regency Hotel, San Francisco, U.S.A.* and the keywords *quiet*, *pizza*, and *cappuccino*.

3.1.2 Standard Query Extensions

These queries extend standard queries by considering additional aspects, e.g., temporal or social aspects. These queries can be used when users have requirements in addition to spatial and textual constraints. Table 1 gives the relations between them and standard queries.

Temporal spatial keyword query [28,113,116]. In addition to the elements of a spatial keyword query, a temporal spatial keyword query has a time predicate. Considering POIs that have an interval-valued attribute that models opening times, Chen et al. [28] study a time-aware Boolean spatial keyword query that takes a query region, a set of query keywords, and a time interval as parameters and returns the top-k POIs that are located in the query region, contain all the query keywords, and have the highest ranking scores. The ranking score takes into account the distance between the POI and the query location and the duration of the overlap between a query time interval and the opening time of the POI. It is defined formally as follows.

Time-aware Boolean spatial keyword query: Consider a set of geo-textual objects D and a query $q = (\rho, \psi, t, r, k)$, where ρ is a query location, ψ is a set of keywords, t is a query time interval, and r is a spatial radius. Let $D(q.\psi)$ be the objects in D containing all the query keywords. The result R of q is a subset of $D(q.\psi)$ that contains k objects such that $\forall o \in R\ (dist(q,o) < q.r \wedge \forall o \in R\ (\forall o' \in (D(q.\psi) \setminus R)\ (\Phi(o) \geq \Phi(o'))))$. Here,

$$\Phi(o) = \frac{|q.t \cap o.t|/|q.t|}{1 + \alpha \cdot dist(q,o)},$$

where $\alpha \in [0,1)$ is a preference parameter that controls the importance of spatial proximity. This query extends a BRSK query with a ranking function that considers temporal overlap and spatial proximity.

Nepomnyachiy et al. [116] study a temporal Boolean range spatial keyword (TBRSK) query. A TBRSK query $q = (\psi, \rho, r, \tau)$ takes four parameters, where ψ is a set of keywords, ρ is a spatial point, r is a query region radius, and τ is a time interval. The result $q(D)$ of q is the subset of D satisfying

$$\forall o \in q(D)(dist(o,q) \leq q.r \wedge q.\psi \subseteq o.\psi \wedge o.t \in q.\tau).$$

This query extends a BRSK query with temporal filtering. Mehta et al. [113] study a variant of the TBRSK query that considers moving geo-textual objects.

Socially-aware spatial keyword query [13, 77, 145]. A socially-aware spatial keyword query integrates social network aspects into spatial keyword querying. Three types of functionality can be distinguished.

(1) *Retrieval of geo-textual objects* [13, 145]: The query $q = (\rho, \psi, k, S)$ takes into account the social relevance of query objects with respect to the query user. Specifically, ρ is a point location, ψ is a set of keywords, k is the number of result objects, and S denotes the social network of the user who issues q. Each object is a triple (ρ, ψ, F), where ρ denotes location, ψ denotes keywords, and F denotes a set of "followers" who show an interest in the object (e.g., liking, sharing, checking into, or recommending the object). The query returns the top-k objects according to a function that takes into account spatial proximity, textual similarity, and social relevance. The social relevance of an object o to a user who issues a query is determined by the relationship between o's followers, F, and the query user's social network, S. The social relevance notion proposed by Wu et al. [145] favors objects having more followers (u) close to the query user u_q in S, while the social relevance notion proposed by Ahuja et al. [13] favors objects having more check-ins by friends of the query user. The ranking functions proposed by Ahuja et al. [13] ($Arank_q(o)$) and Wu et al. ($Wrank_q(o)$) [145] are defined as follows, by extending Eq. 1 and 2, respectively.

$$Arank_q(o) = \alpha \cdot ss(q,o) + \beta \cdot st(q,o) + \gamma \cdot sd(q,o),$$

where $\alpha + \beta + \gamma = 1$.

$$Wrank_q(o) = \frac{ss(q,o)}{st(q,o) \cdot sd(q,o)},$$

$$sd(q,o) = 1 + \sum_{u \in o.F} \alpha^{mindist(u_q,u)},$$

where $\alpha \in [0,1)$ is a damping factor and $mindist(u_q,u)$ denotes the length of the shortest path between u_q and

Query Types	Acronyms	Relation to Standard Queries
Boolean range SK query [41,45,49, 68,83,102,135,163,177]	BRSK	n.a.
Boolean kNN SK query [25,76,98, 133,146,158,163]	BkSK	n.a.
Top-k kNN SK query [48, 59, 91, 122,152,153,158,163]	TkSK	n.a.
Temporal SK query [28,113,116]	TBRSK	Add temporal filtering or time-based ranking to the BRSK query.
Socially-aware SK query [13, 77, 145]	TkLUS	*Retrieval of objects*: Extend the TkSK query to consider social aspects in the ranking function. *Retrieval of users*: Extend the TkSK query to find top-k users who create geo-tagged posts that satisfy an SK constraint. *Retrieval of terms*: Extend the BRSK query to find the top-k frequent terms in the geo-tagged posts of the social network friends of a query user.
Direction-aware SK query [84]	n.a.	Extend the BkSK query to consider an extra direction constraint.
Preference-aware SK query [15,88, 134]	TkSKP, LGP	Extend the TkSK query to consider a set of nearby POIs in the ranking function.
Top-k prestige-based SK query [23]	TkPSK	Extend the TkSK query to use a ranking function that combines linearly prestige-based relevance (PR) and spatial proximity. PR captures the textual relevance to a query of an object and its nearby objects.
Moving SK query [74,142,147,148]	MTkSK	This query can be regarded as a moving version of a TkSK query.
Reverse SK query [55, 58, 95–97, 171]	RBkSK, RTkSK	Given a query object q, this query finds the set of objects whose BkSK or TkSK query result includes q.
SK skyline query [86,121,128]	n.a.	This query finds the set of objects that are not SK dominated by any other object. The computation of SK domination is similar to the ranking function of a BkSK or TkSK query.

Table 1 Relations between Standard Query Extensions and Standard Queries (SK Denotes Spatial Keyword)

u in social network S. As the two queries extend the TkSK query to cover social aspects, we refer to them as Social_TkSK.

(2) *Retrieval of users* [13, 77]: Given a query point location or region and a set of query keywords, this kind of query returns the top-k users, taking into account their proximity to the query location, the textual similarity between their profile and the query keywords, and, possibly, the social connectivity among the users. In particular, Jiang et al. [77] assume a social network S and let U denote the users in S and let O denote the geo-tagged posts made by users in U. In this setting, they find the top-k users who have made geo-tagged posts relevant to the query keywords within a query region. The top-k local user search (TkLUS) query is given by $q = (\psi, \rho, r)$, where ψ is a set of keywords, ρ is a location, and r is a radius. The query is defined as follows.

TkLUS query: Query $q = (\psi, \rho, r)$ finds a set $U_q \subseteq U$ of (at most) k users such that (1) $\forall u \in U_q$ $(\exists o \in O_u$ $(dist(q, o) \leq q.r \wedge q.\psi \cap o.\psi \neq \emptyset))$, where $O_u \subseteq O$ and denotes the set of posts made by u, and (2) $\forall u \in U_q$ $(\forall u' \in (U \setminus U_q)$ $(score(u', q) \leq score(u, q)))$. Here, $score(\cdot, \cdot)$ computes the relevance of a user to a query, taking into account both spatial proximity and textual similarity. This query extends the TkSK query to user retrieval.

Ahuja et al. [13] propose to find the top-k users based on a scoring function that combines linearly (1) the spatial proximity between the user location and the query location, (2) the textual similarity between the user profile and query keywords, and (3) the cardinality of the set of friends of the user. This query extends the TkSK query.

(3) *Retrieval of terms* [13]: Given a query region, a query user, and the user's friends, this type of query returns terms with the highest frequency in the geo-tagged posts that belong to the query region and are posted by friends. This query can be used to discover trending topics among friends in a geographic region.

Direction-aware spatial keyword query [84]. In addition to the spatial and textual predicates, a direction-aware spatial keyword query q contains a direction constraint $[\alpha, \beta]$, which captures that the user is only interested in geo-textual objects with directions from q in $[\alpha, \beta]$. Thus, the query is given by $q = (\rho, \psi, k, [\alpha, \beta])$, where ρ, ψ, and k denotes a location, a keyword set, and the result cardinality, respectively, and $[\alpha, \beta]$ represents the query direction. Given a set of geo-textual objects D, let $D(q.\psi)$ denote the subset of D that contain all keywords in $q.\psi$. Query q finds a set R of (at most) k objects in $D(q.\psi)$ such that (1) $\forall o \in R(\forall o' \in (D(q.\psi) \setminus R)(dist(o, q) \leq dist(o', q)))$, and (2) $\forall o \in R(\alpha \leq direc(o, q) \leq \beta)$, where $direc(o, q)$

denotes the direction of o from q. We could even just say that this query takes one more query parameter than the BkSK query and adds (2) to the definition of the BkSK query. Then we have defined the query unambiguously.

Preference-aware spatial keyword query [15, 88, 134]. The rationale behind this type of query is that users are usually interested in geo-textual objects based on the quality of other *feature objects* (i.e., facilities) that are located in their neighborhoods. Feature objects are typically described by non-spatial attributes such as quality or rating, in addition to the textual description.

Given a set of geo-textual objects O of interest and a set of feature geo-textual objects F, the top-k spatial keyword preference (TkSKP) query takes the form $q = (\psi, r, k)$, where ψ is a keyword set, r is a spatial selection criterion (normally a radius), k indicates the number of results. It retrieves (at most) k objects based on a score that combines the textual relevance score and the non-spatial score of feature objects in their neighborhood [15, 134]. In particular, Tsatsanifos et al. [134] define the spatial keyword preference score based on a predefined score for each type of feature objects. Assume that there are m types of feature objects $\{F_1, F_2, \cdots, F_m\}$ and each $F_i (1 \leq i \leq m)$ comprises a specific type of feature objects, the preference score of a feature object $t \in F_i$ for the query keywords is computed by Eq. 3.

$$s(t) = (1 - \lambda) \cdot t.s + \lambda \cdot st(t, q), \tag{3}$$

where $\lambda \in [0, 1]$ is a preference parameter and $t.s$ is the non-spatial score of t. Eq. 4 computes the preference score of an object o for F_i.

$$\tau_i(o) = max\{s(t) | t \in F_i \wedge dist(o, t) \leq q.r \wedge st(t, q) > 0\}, \tag{4}$$

where $q.r$ is a query radius. Then the overall spatial keyword preference score of an object o is computed by Eq. 5:

$$\tau(o) = \sum_{1 \leq i \leq m} \tau_i(o). \tag{5}$$

Almeida et al. [15] study an alternative definition of the score. Their score pre-defines a set of objects as feature objects. Then the spatial keyword preference score of a data object o is defined based on the textual similarity between the feature objects close to o and the query keywords (Eq. 6).

$$\tau(q, o) = max\{st(q, t) \,|\, t \in F \wedge dist(o, t) \leq q.r\}, \tag{6}$$

where $st(q, t)$ denotes the textual similarity between q and feature object t, and F denotes the set of feature objects.

Li et al. [88] define a different preference-aware query, the location-aware group preference (LGP) query. The LGP query can be used when a group of users want to find a destination POI labeled with a specific category feature (e.g., hotel) and each user has a preference about the POIs near the destination POI. We introduce the location-aware preference (LP) query and then explain the LGP query. An LP query is given by $q = (\rho, f_d, \Psi)$, where ρ denotes a location, f_d denotes the category feature of the result object, and $\Psi = \{f_0, f_1, \cdots, f_n\}$ denotes the set of category features that the user wants the objects near the result object to belong to. The LP query q returns an object o_q belonging to f_d that has the largest score computed using Eq. 7:

$$\begin{aligned}\tau(q, o) = \lambda \cdot (1 - \frac{dist(q, o)}{dist_{max}}) + (1 - \lambda) \cdot \frac{1}{|q.\Psi|} \\ \cdot \sum_{f_i \in q.\Psi} (1 - \frac{minDist(o, f_i)}{dist_{max}}),\end{aligned} \tag{7}$$

$\lambda \in [0, 1]$ is a preference parameter and $minDist(o, f_i)$ is a function returning the minimum distance between o and the objects belonging to f_i.

An LGP query $Q = \{q_0, q_1, \cdots, q_m\}$ consists of a set of LP queries $q_0, \cdots, q_m$ that all have the same category feature of the result object (i.e., $\forall q_i, q_j \in Q$ $(q_i.f_d = q_j.f_d)$). The LGP query Q returns the object that has the largest score computed by $\sum_{q_i \in Q} \tau(q, o)$.

Top-k prestige-based spatial keyword (TkPSK) query [23]. A TkPSK query $q = (\rho, \psi)$ takes two parameters: ρ that denotes a location and ψ that denotes a keyword set. It retrieves (at most) k geo-textual objects ranked according to a scoring function that combines linearly prestige-based relevance (PR) (i.e., $pr(q, o)$) and spatial proximity (i.e., $dist(q, o)$).

In particular, PR captures both the textual relevance between o and q and the textual relevance between q and objects near o. To compute PR, a weighted and undirected graph $G = (V, E)$ is introduced. Each vertex in V corresponds to a geo-textual object. An edge exists between a pair of objects o_i and o_j iff $dist(o_i, o_j) \leq \gamma$ and $sim(o_i, o_j) \geq \xi$, where γ and ξ are given thresholds. The weight of edge (o_i, o_j) is $dist(o_i, o_j)$. Eq. 8 computes the value of PR (vector $\mathbf{p}$):

$$\begin{aligned}&\mathbf{p} = (1 - \alpha)\mathbf{C^T}\mathbf{p} + \alpha \mathbf{u_Q}, \\ &\mathbf{u_Q} = [v_1, ..., v_{|D|}]^T, v_i = sim(q, o_i), 1 \leq i \leq |D|,\end{aligned} \tag{8}$$

where $\mathbf{C^T}$ is the normalized adjacency matrix of graph G such that $\sum_{j \in V} C(a, b) = 1$, where $C(a, b)$ represents the normalized weight of edge (a, b); and column vector $\mathbf{u_Q}$ is the initial PR vector in which each element is the similarity between an object and the query. Parameter α represents the probability of a random surfer jumping to the set of initially relevant objects ($v_i > 0$) rather than following the edges in the graph. This parameter can be used to balance the relevance of an object and the influence of its relevant neighbors, i.e., the parameter allows for tuning according to user-specific requirements. In particular, a smaller α favors objects with nearby relevant objects, while a larger α favors objects with high initial PR scores. This query extends the TkSK query by considering the influence of nearby objects.

Moving spatial keyword query [74,142,147,148]. A moving spatial keyword query is a continuous query that takes a continuously moving spatial location and a set of keywords as parameters. This query enables a mobile user to be kept continuously aware of relevant near-by objects as the user moves. For example, a tourist visiting New York City may issue a "lunch special vegetarian" query to be alerted about nearby opportunities for lunch, and individuals looking for entertainment may issue a "happy hour free snacks" query in the late afternoon to be alerted about bars with happy hour deals with free snacks. The moving top-k spatial keyword (MTkSK) query has been studied, which can be regarded as the moving version of the TkSK query.

The MTkSK query [74, 142, 147, 148] is given by $q = (\rho, \psi, k)$, where ρ denotes a point location, ψ denotes a keyword set, and k denotes the cardinality of the result. As the query is moving, for each new location ρ', its result is $\langle A, R\rangle$, where A is the top-k result of $q' = (\rho', \psi, k)$ and R is the corresponding safe region. The safe region of q' indicates a region in which the top-k result of q' remains unchanged, which is denoted by $R = \{\rho \in S \,|\, \forall o \in O(\forall o' \in (O \setminus A)(score(q, o) \leq score(q, o')))\}$, where S denotes the global spatial space. The scoring function considers the spatial proximity and textual similarity (Eq. 1 and Eq. 2).

Reverse spatial keyword (RSK) query [55, 58, 95–97, 171]. The RSK query comes in two variants: monochromatic and bichromatic reverse spatial keyword query.

For the monochromatic RSK query, the data and query objects are of the same type. In particular, given a set of data objects D and a query object $q \in D$, the monochromatic RSK query finds objects in D whose top-k most "similar" objects include q, where the similarity metric combines the spatial proximity and textual similarity. For the bichromatic RSK query, data objects and query objects are of different types. Specifically, given a set of data objects D, a set of query objects Q, and a query object $q \in Q$, a bichromatic RSK query finds objects in D whose top-k most "similar" objects in Q include q, where the similarity metric again combines the spatial proximity and textual similarity.

Each variant can be further classified into two types: reverse Boolean kNN spatial keyword (RBkSK) query and reverse top-k kNN spatial keyword (RTkSK) query. The four types of reverse spatial keyword queries are defined as follows.

Monochromatic RBkSK query [55, 58, 171]: Given a set of data objects D and a query object $q \in D$, the monochromatic RBkSK query retrieves objects whose BkSK query results include q, i.e.,

$$\text{RB}k\text{SK}(q, k, D) = \{o \in D \,|\, q \in \text{B}k\text{SK}(o.\psi, o.\rho, k)\}. \quad (9)$$

Monochromatic RTkSK query [95–97]: Given a set of data objects D and a query object $q \in D$, the monochromatic RTkSK query retrieves objects whose TkSK query results include q, i.e.,

$$\text{RT}k\text{SK}(q, k, D) = \{o \in D \,|\, q \in \text{T}k\text{SK}(o.\psi, o.\rho, k)\}. \quad (10)$$

Bichromatic RBkSK query [171]: Given a set of data objects D, a set of query objects Q, and a query object $q \in Q$, the bichromatic RBkSK query retrieves objects from D whose BkSK query results among Q include q, i.e.,

$$\begin{aligned}\text{RB}k\text{SK}(q, k, D, Q) = \{o \in D \,|\, q \in Q \wedge \\ q \in \text{B}k\text{SK}(o.\psi, o.\rho, k)\}.\end{aligned} \quad (11)$$

Bichromatic RTkSK query [44,96]: Given a set of data objects D, a set of query objects Q, and a query object $q \in Q$, the bichromatic RTkSK query retrieves objects from D whose TkSK query results among Q include q, i.e.,

$$\begin{aligned}\text{RT}k\text{SK}(q, k, D, Q) = \{o \in D \,|\, q \in Q \wedge \\ q \in \text{T}k\text{SK}(o.\psi, o.\rho, k)\}.\end{aligned} \quad (12)$$

Spatial keyword skyline query [86, 121, 128]. A spatial keyword skyline query q takes as parameters a keyword set, a (optional) query region, and one or more spatial points. The result comprises those objects that are not *dominated* by any other objects. An object dominates another one only if it is as good as or better in all dimensions and better in at least one dimension.

Existing studies [86,121] on the spatial keyword skyline query define the "dimensions" in terms of textual similarity and spatial proximity. Li et al. [86] decompose the textual similarity and regard the similarity (measured by edit distance) in terms of each keyword as a dimension, yielding the *edit-distance-based spatial keyword skyline*. Regalado et al. [121] consider the textual similarity as a single dimension, which is called the *text-similarity-based spatial keyword skyline*. The formal definitions are presented as follows.

Edit-distance-based spatial keyword skyline: Given a set of geo-textual objects D and a query object $q \in D$, the result R of q is those objects in D that are not dominated by any other object located in the query region. An object o dominates another object o' if the maximum edit distance between each keyword in q and the keywords in o is no larger than the maximum edit distance between each keyword in q and the keywords in o'.

Text-similarity-based spatial keyword skyline: Given a set of geo-textual objects D and a query object $q \in D$, the result R of q is those objects in D that are not dominated by any other object in D located in the query region. An object o dominates another object o' only if the distance between o and q is no larger than the distance between o' and q, and the textual similarity between o and q is no smaller than the similarity between o' and q.

Shi et al. [128] consider a skyline query that can accommodate a set of spatial points as a parameter. They compute the spatial-textual relevance score using Eq. 2 and regard the spatial-textual relevance score between an object o and each spatial point in q as one dimension. Specifically, they develop a spatio-textual dominance (STD) model to define the spatial keyword skyline query.

STD model: An object o spatio-textually dominates another object o' iff $\forall q_i \in Q\,(ST(o, q_i) \geq ST(o', q_i))$ and $\exists q_i \in Q\,(ST(o, q_i) > ST(o', q_i))$.

STD spatial keyword skyline: Given a set of query objects $Q = \{q_0, q_1, ..., q_n\}$ and a set of data objects D, the results D of Q consists of the objects in D that are not spatio-textually dominated by any other object in D.

3.1.3 Group Queries

Group queries consider inter-object relations and find sets of objects that answer the particular query collectively. Table 2 gives a brief introduction to each type of group queries.

Group spatial keyword query [22, 24, 26, 27, 43, 50, 64, 94, 110, 130, 161, 162, 164]. This query finds a group of geo-textual objects G that covers the query keywords collectively (i.e., $\forall w \in q.\psi\,(\exists o \in G\,(w \in o.\psi)))$ and optimizes a predefined cost function.

We classify existing studies into two categories. The first encompasses queries that optimize a cost function that considers inter-object distance and the distance between objects and the query location. The second encompasses queries that consider a cost function that considers additional aspects beyond distance, e.g., user ratings of keywords.

Existing studies in the first category pursue five optimization goals.

(1) Minimizing the maximum distance between any pair of objects in G, as done in the m-Closest Keywords (mCK) query [64, 161, 162]. Choi et al. [43] introduce a variant of the mCK query that aims to minimize the product of $(|G|-1)$ and the maximum distance between any pair of objects in G.

(2) Minimizing the sum of the distances between each object in G and the query (i.e., the SUM-GSK query [22, 24, 69]).

(3) Minimizing a cost function that combines linearly the maximum distance between any pair of objects in G and the maximum distance between an object in G and q (i.e., the MAX+MAX GSK query [22, 24, 94]). Additionally, one study [94] considers a variant of the MAX+MAX GSK query, where the score function computes the maximum distance between any pair of objects in $G \cup \{q\}$.

(4) Minimizing a cost function that combines linearly the minimum distance between any pair of objects in G and the maximum distance between an object in G and q (i.e., the MIN+MAX GSK query [22]).

(5) Minimizing a generalized cost function that can be instantiated to the above four types of cost functions (i.e., generalized GSK query [27]).

Given a collection of geo-textual objects D, we present the definition of each group spatial keyword query.

mCK query: An mCK query q takes m keywords as a parameter. It finds a group of objects $G \subseteq D$, each of which contains at least one query keyword, such that $\cup_{o \in G} o.\psi \supseteq q$ and such that the maximum distance between any pair of objects in G is minimized.

SUM-GSK query: A SUM-GSK query $q = (\rho, \psi)$ takes a spatial point ρ and a set of keywords ψ as parameters. It finds a group of objects $G \subseteq D$ such that $\cup_{o \in G} o.\psi \supseteq q.\psi$ and the sum of the distances between each object in G and $q.\rho$ is minimized.

Query Types	**Description**
Group SK query [22, 24, 26, 27, 43, 50, 64, 94, 110, 130, 161, 162, 164]	It finds a group of objects that cover the query keywords collectively and that minimize a cost function considering the sum of the inter-object distances.
Clue-based SK search [92]	It takes as input a target object and so-called spatio-textual clues that describe nearby objects and the spatial relations between them and the target object. It retrieves a set of objects that have the highest SK similarity with the spatio-textual clues.
Spatial pattern matching [56]	It finds a set of objects that satisfy a spatial pattern collectively. A spatial pattern describes the spatial relations between the objects and the keyword constraint for each object.
Top-k spatial textual clusters query [130, 144]	It finds top-k clusters of objects that are closed to the query location and that are textually relevant to the query keywords.

Table 2 Summary of Group Queries (SK Denotes Spatial Keyword)

MAX+MAX GSK query: A MAX+MAX GSK query $q = (\rho, \psi)$ takes a spatial point ρ and a keyword set ψ as parameters. It finds a group of objects $G \subseteq D$ such that $\cup_{o \in G} o.\psi \supseteq q.\psi$ and the following cost function is minimized:

$$\alpha \cdot max_{o \in G}(dist(o, q)) + (1-\alpha) \cdot max_{o_1, o_2 \in G}(dist(o_1, o_2)).$$

MIN+MAX GSK query: A query $q = (\rho, \psi)$ takes a spatial point ρ and a keyword set ψ as parameters. It finds a group of objects $G \subseteq D$ such that $\cup_{o \in G} o.\psi \supseteq q.\psi$ and the following cost function is minimized:

$$\alpha \cdot min_{o \in G}(dist(o, q)) + (1-\alpha) \cdot max_{o_1, o_2 \in G}(dist(o_1, o_2)).$$

In addition, the SUM-GSK query, the MAX+MAX GSK query, and the MIN+MAX GSK query are extended to their corresponding top-k version [22].

Generalized GSK query: A query $q = (\rho, \psi)$ again takes a spatial point ρ and a keyword set ψ as parameters. It returns a group of objects $G \subseteq D$, such that $\cup_{o \in G} o.\psi \supseteq q.\psi$ and $cost(G)$ is minimized. Function $cost(G)$ includes a distance component that is defined as follows.

$$\tau(G, \phi_1) = [\sum_{o \in G} (dist(o, q))^{\phi_1}]^{\frac{1}{\phi_1}},$$

where $\phi_1 \in \{1, \infty, -\infty\}$ is a user-provided parameter corresponding to the summation, the maximum, or the minimum of the distance between the objects in G and the query location, respectively. Specifically,

$$\tau(G, \phi_1) = \begin{cases} \sum_{o \in G} dist(o, q), & \text{if } \phi_1 = 1 \\ \max_{o \in G} dist(o, q), & \text{if } \phi_1 = \infty \\ \min_{o \in G} dist(o, q), & \text{if } \phi_1 = -\infty \end{cases}$$

Function $cost(G)$ is defined as follows.

$$cost(G, \phi_1, \phi_2) =$$
$$\{[\alpha \cdot \tau(G, \phi_1)]^{\phi_2} + [(1-\alpha) \cdot \max_{o_1, o_2 \in G} dist(o_1, o_2)]^{\phi_2}\}^{\frac{1}{\phi_2}},$$

where $\alpha \in (0, 1)$ and $\phi_1 \in \{1, \infty, -\infty\}$ and $\phi_2 \in \{1, \infty\}$ are user-provided parameters.

Next, the second category of studies includes queries that optimize a cost function involving additional aspects beyond distance. We still denote a collection of geo-textual objects by D.

Inherent-cost aware GSK query: This query [26] assumes that each object is associated with an inherent cost (e.g., the entrance fee of a POI). It aims to find a group of objects that collectively cover the query keywords and have the smallest cost. Formally, the inherent-cost aware GSK query $q = (\rho, \psi)$ takes two parameters: ρ denotes the query location and ψ denotes a keyword set. It finds a group of objects $G \subseteq D$ such that $\cup_{o \in G} o.\psi \supseteq q.\psi$ and the following function is minimized:

$$\max_{o \in G} dist(o, q) \cdot \sum_{o \in G} o.cost,$$

where $o.cost$ denotes the inherent cost of o.

Level-aware GSK query: This query [164] assumes that each object has a level vector that assigns an integer value to each keyword. For example, a POI may have multiple features, each of which has a score that expresses its goodness. Additionally, each object has a cost. The level-aware GSK query retrieves a group of objects that has the minimum cost and satisfies a level requirement for each query keyword. Formally, each object $o = (\rho, \psi, V, \beta)$ has four attributes: ρ denotes a spatial location, ψ denotes a set of keywords, V denotes a level vector that assigns an integer value to each keyword in ψ, and β denotes the cost of o. A level-aware GSK query $q = (\rho, \psi, W, \theta)$ takes four parameters: ρ denotes the query location, ψ denotes a set of keywords, W denotes a normalized weight vector whose cardinality equals the maximum element value of the level vector, and $q.\theta$ denotes a threshold. The query finds a group of objects $G \subseteq D$ such that $cost(G)$ is minimized and $\forall w \in q.\psi$ $(cov(G, w) \geq q.\theta)$. Here,

$cost(G) = \sum_{o \in G}(o.\beta \cdot dist(o,q))$, and $cov(G,w) = \sum_{o \in G} q.W[o.V[w]]$.

Best keyword cover query: The best keyword cover (BKC) query [50] assumes that each object has a rating. The query finds a group of objects that cover the query keywords and have the maximum score. A group of objects G covers a set of keywords ψ if each object in G covers exactly one keyword in ψ that is not covered by another object. Formally, a BKC query $q = (\rho, \psi)$ takes two parameters: a spatial location ρ and a keyword set ψ. It finds a group of objects $G \subseteq D$ that covers $q.\psi$ and maximizes the following function:

$$\alpha \cdot \left(1 - \frac{\max_{o_i, o_j \in G} dist(o_i, o_j)}{maxDist}\right) + (1-\alpha) \cdot \frac{\min_{o \in G} o.rating}{maxRating},$$

where $\alpha \in [0,1]$ is a user-provided parameter.

In addition, Mahmood et al. [110] propose an SQL extension to express different types of group spatial keyword queries. Spatial and textual building-block operators and predicates are provided for this purpose.

Clue-based spatial keyword search [92]. To handle the case where a user cannot provide exact spatial and textual information in a spatial keyword query, Liu et al. [92] propose a clue-based spatial keyword query that allows a user to input spatio-textual clues for the retrieval of geo-textual objects. Such clues describe the spatio-textual context of a target object, which includes nearby objects and the spatial relations between them and the target object.

Specifically, let D be a set of geo-textual objects with category information. A clue is specified in terms of categories of objects near the target object (called clue objects) and the spatial relations (i.e., distance and direction) between the clue objects and the target object. A clue-based spatial keyword query is given by $q = (o_q, r, N, E)$, where $o_q \in D$ represents a target geo-textual object, r is a query region, N is a set of clue objects, and E represents a set of edges indicating the connection between objects in N. The query retrieves (at most) k objects from D that have the same keywords/category as o_q and have the highest *spatio-textual context similarity* with o_q. The spatio-textual context similarity between o_q and o_i, where $o_i \in D$, is defined as the extent to which o_i and o_q have similar spatio-textual contexts based on a matching between N and objects near o_i with the same textual information in terms of the distances and directions specified in E.

Spatial pattern matching [56]. Fang et al. [56] study the following spatial pattern matching problem. Given a set of geo-textual objects O and a spatial pattern P, they aim to find all subsets of O that match P. A spatial pattern P is represented by a graph (V, E), where V is the set of vertices and E is the set of edges. Each vertex $v_i \in V$ has a keyword w_i. Each edge $(v_i, v_j) \in E$ has a distance interval $[l_{i,j}, u_{i,j}]$ and one of the signs: $v_i \rightarrow v_j$, $v_i \leftarrow v_j$, $v_i \leftrightarrow v_j$, and $v_i - v_j$. Two objects o_i and o_j form an e-match of edge (v_i, v_j) if o_i contains w_i, o_j contains w_j, and the distance between o_i and o_j satisfies the constraint specified by $[l_{i,j}, u_{i,j}]$ and the sign on edge (v_i, v_j). The meanings of the signs are as follows:

- $v_i \rightarrow v_j$: There exists no object in O with keyword w_j having a distance less than $l_{i,j}$ from o_i.
- $v_i \leftarrow v_j$: There exists no object in O with keyword w_i having a distance less than $l_{i,j}$ from o_j.
- $v_i \leftrightarrow v_j$: There exists no object in O with keyword w_j (resp. w_i) having a distance less than $l_{i,j}$ from o_i (resp. o_j).
- $v_i - v_j$: Allows the existence of objects besides o_i and o_j in O having a distance less than $l_{i,j}$.

A set of objects S ($S \subseteq O$) matches P if (1) there exists an e-match in S for each edge of P, and (2) S is minimal, i.e., no subset S' of S satisfies that for each edge of P, there exists an e-match in S'. Given a spatial pattern P and a collection of geo-textual objects O, the spatial pattern matching problem is to find all $S \subseteq O$ that match P.

Top-k spatial textual clusters query [130, 144]. Wu and Jensen [144] investigate the problem of retrieving top-k spatial textual clusters. Specifically, a top-k spatial textual clusters (k-STC) query returns k clusters that (i) are closest to the query location, (ii) contain objects that have at least one query keyword, and (iii) have a density that exceeds a threshold. Skovsgaard and Jensen [130] propose a top-k groups spatial keyword query that returns k disjoint groups of objects. The query takes a location and a set of query keywords as parameters and returns k groups of objects such that (i) the objects in a group are close to each other, (ii) the group is close to the query location, and (iii) the objects in the group are textually similar to the query keywords. The ranking score of a group is computed by considering the textual relevance of the group of objects to the query keyword and the distance between the group and the query as well as the diameter of the group.

3.1.4 (Top-k) Spatio-Textual join

The join takes two sets of geo-textual objects $R = \{r_0, r_1, ...\}$ and $S = \{s_0, s_1, ...\}$, where R and S can be either different or the same set, as arguments. A spatio-textual join [19, 54, 75, 93, 120, 166] finds all pairs of objects from R and S that are both spatially close and textually similar. Formally, given a spatial distance threshold θ_s and a textual similarity threshold θ_t, a spatio-textual join retrieves all pairs (x, y) with $x \in R$ and $y \in S$, such that $dist(x, y) \leq \theta_s$ and $sim(x, y) \geq \theta_t$. In some proposals [19,120,166], each object is assumed to have a point location, and the spatial distance is defined as the Euclidean distance, while in other proposals [54,93], each object is assumed to have a region, and the spatial distance is defined as the overlap between two objects. Next, a top-k spatio-textual join [75] finds top-k pairs based on a scoring function that combines linearly the spatial distance and textual similarity.

3.1.5 Region Finding/Analysis

Region finding/analysis queries find sets of regions satisfying pre-defined constraints or investigate data distributions in regions. Table 3 gives a brief introduction to each type of region finding/analysis queries.

Geosocial search [117]. Here, the objective is to find geographical regions based on geo-tagged social network posts. A geosocial search query is given by $q = (\psi, \tau, k)$, where ψ is a set of keywords, τ is a time interval, and k is an integer. A geo-tagged post is considered relevant to q if its textual content contains ψ and its timestamp is within τ. Let μ be a function that computes the score of a set of geo-tagged posts within a region with respect to q. Given a set of regions Q, a geosocial query q finds top-k disjoint regions R with the highest score, i.e.,

$$\begin{aligned}\{R \mid R \subseteq Q \wedge |R| = k \wedge (\forall r \in R, \overline{r} \in (Q \setminus R))\\ (\mu(r, q) \geq \mu(\overline{r}, q))\\ \wedge(\forall r, r' \in R)(\varDelta(r, r') = 0)\},\end{aligned}$$

where function $\varDelta$ computes the area of the intersection between two regions.

Reverse top-k keyword-based location query [149]. This query is based on the TkSK query. A Reverse_TkSK query $q = (\psi, o, k)$ takes as parameters a set of keywords ψ, an object o, and an integer k. It retrieves the maximum spatial region r such that for any TkSK query $q' = (p, \psi, k)$ with $p \in r$ and the same parameters ψ and k as q, o is inside the result of q'. Formally,

$$r = \{p \in \varOmega \mid o \in S(q') \wedge o \in O\},$$

where $\varOmega$ is the region in which the geo-textual objects reside and $S(q')$ is the result of the TkSK query $q' = (p, \psi, k)$.

Top-k most frequent terms query [12]. This query finds the top-k most frequent terms given a region and a time interval. Let D be a set of geo-textual objects with a time attribute. The count of a term w for a set of objects S is the number of objects in S whose textual information contains w. A top-k most frequent terms query is given by $q = (R, T, k)$, where R denotes a spatial region, T denotes a time interval $[t_s, t_e]$, and k indicates the number of results. It returns the top-k most frequent terms from objects in D that fall in the spatial region and time interval.

Topic exploration [169, 170]. Zhao et al. [169] propose to explore topics over geo-textual objects within a specified region and time interval. This query, called Topic_RT, takes as parameters a rectangular region R and a time interval T. It returns k topics from the geo-textual objects whose locations fall into R and whose timestamps fall into T. The topic assignment of a geo-textual object is based on the latent dirichlet allocation (LDA) model, a commonly used topic model. In follow-on work [170], the Topic_RT query is extended to capture the market competition of different brands over each topic for a category of business (e.g., coffeehouses).

3.1.6 Query Modification

Recall that the TkSK query takes keywords, k, and a value that balances the importance of spatial proximity versus textual similarity as parameters. This information can be difficult for users to specify. Query modification investigates how to refine queries so that users are better served by the results. Two types of such queries has been considered: why-not spatial keyword queries and why-not group spatial keyword queries. Why-not spatial keyword queries investigate how to refine a TkSK (or TkSK-based) query based on a set of objects that the user expect to appear in the query result. Why-not group spatial keyword queries extend the why-not spatial keyword queries to consider a group of TkSK (or TkSK-based) queries.

Why-not spatial keyword query [33, 34, 38, 39, 176]. To provide users with better results, Chen et al. [34,38,39] study the problem of modifying the keywords, the value of k, and the preference parameter α in a TkSK query. The idea is that a user receives a result set and finds that one or more objects expected to be in the result are missing. This signals that the parameters may be set inappropriately. The challenge is then how to modify them "minimally" so that the result includes the missing objects.

Specifically, given a set of geo-textual objects D, a missing object set M, and an original TkSK query q, a why-not query returns a TkSK query q' with revised query keywords and k [34] or revised α and k [39] such

Query Types	Description
Geosocial search [117]	It finds top-k disjoint regions based on a ranking function that considers the SK similarity between the objects in the regions and the query.
Reverse top-k keyword-based location query [149]	It is based on the TkSK query. Given a query object, it finds the maximum spatial region such that the result of any TkSK query within that region includes the query object.
Top-k most frequent terms query [12]	It finds the top-k most frequent terms of objects in a given region and time interval.
Topic exploration [169, 170]	It explores topics over a set of objects within a specified region and time interval. The topic assignment of each object is based on the LDA model.

Table 3 Summary of Region Finding/Analysis Queries (SK Denotes Spatial Keyword)

that the result of q' contains the objects in M. A function is introduced that calculates the penalty costs of q' as a refinement of q. Then the problem is to find the q' with minimum penalty cost. A subsequent study [33] aims to refine the query parameters of a direction-aware spatial keyword query. In particular, it returns a refined query with revised k and direction range.

Zheng et al. [176] focus on finding users' most preferable geo-textual objects by interacting with users in multiple rounds. The query considered is basically a TkSK query, where each query keyword has a user preference ranging from 0 to 1. The user's preferences for all query keywords can be represented by a preference vector $\mathbf{w}$, which is merged into the scoring function of the TkSK query. At first, a user issues a preference-aware TkSK query with an initial preference vector $\mathbf{w}$. Then the query is processed in rounds. In each round, the system returns at most K tuples, and the user picks their favorite tuple. This enables adjustment of the user's preferences to become $\mathbf{w}'$. At the end, a final set of k objects is returned based on $\mathbf{w}'$.

Why-not group spatial keyword query [174]. Zheng et al. [174] investigate the problem of refining top-k group spatial keyword queries. Let O denote a set of geo-textual objects, where each object o has a location ρ and a set of keywords ψ. Assume that there are m users and that each user u has a weight of distance tolerance w and a location ρ. A top-k group spatial keyword (top-k GSK) query is given by $q = (Q, \mathbf{w}, t_q, k)$, where Q denotes a set of user locations, $\mathbf{w}$ denotes a weight vector of user tolerances to distance, t_q is a query keyword, and k is an integer. The query returns up to k objects that contain t_q and have the highest scores. The score of an object is a linear combination of $\mathbf{w}$ and the normalized Euclidean distance between user locations and the object.

The complexity of the top-k GSK query makes it difficult for users to choose appropriate parameters. As before, users may find that a set M of objects that were expected to be in a query result are missing. The why-not group spatial keyword (WGSK) query returns a refined top-k GSK query with the smallest modification penalty that contains the objects in M. Let k' (resp. k_o) denote the number of objects to be returned in the refined (resp. original) query, and $\mathbf{w}'$ (resp. $\mathbf{w_o}$) denote the weight vector in the refined (original) query. The modification penalty is based on Δk, $\Delta \mathbf{w}$, and the rank of missing objects in the original query, where $\Delta k = \max(0, k' - k_o)$ and $\Delta \mathbf{w} = ||\mathbf{w}' - \mathbf{w_o}||_2$. Further, for a missing object with rank r_o in the original query, the modification penalty is computed as $\lambda \cdot \frac{\Delta k}{r_o - k_o} + (1-\lambda) \cdot \frac{\Delta \mathbf{w}}{\sqrt{1+\sum w_o[i]^2}}$, where $\lambda \in (0,1)$ is a predefined parameter.

3.2 Querying over Road Networks

This line of work investigates the problems of querying geo-textual data embedded in a road network. Here, the spatial distance between two objects is the network distance. We denote a road network by $G = (V, E, W)$, where V is the set of vertices, E is the set of edges, and W is a set of weights (network distance) associated with edges. Depending on the setting, geo-textual objects may be assigned to the vertices or edges of G. Although similar queries exist in querying in Euclidean space, the proposed solutions have noticeable differences. Table 4 introduce the queries and their relations to queries in Euclidean space.

3.2.1 Standard Spatial Keyword Queries in Road Networks

A Boolean range spatial keyword (BRSK$_{\text{road}}$) query [101] and a top-k kNN spatial keyword (TkSK$_{\text{road}}$) query [123] extend a BRSK query and a TkSK query, respectively, to road networks. The problem definitions are equivalent except in how spatial relations are computed. To avoid repetition, we thus skip their problem definitions.

3.2.2 Extension of Standard Spatial Keyword Queries

Moving spatial keyword query [62, 150, 175]. A moving spatial keyword query takes a continuously

Query Name	Query Category	Counterpart	Description
Boolean range SK (BRSK$_{\text{road}}$) query [101]	Standard SK queries	RSK query	Finds the set of objects that contain the query keywords and that are within a query range.
Top-k kNN SK (TkSK$_{\text{road}}$) query [123]	Standard SK queries	TkSK query	Finds k objects ranked the highest according to network distance and textual relevance.
Moving Boolean kNN SK (MBkSK$_{\text{road}}$) query [62,150,175]	Extensions of standard SK queries	n.a.	Maintains continuously the k nearest objects to the moving query that contain the query keywords.
Moving top-k SK (MTkSK$_{\text{road}}$) query [62, 150]	Extensions of standard SK queries	MTkSK query	Maintains continuously k objects sorted in ascending order of a score considering spatial proximity and textual similarity.
Reverse SK (RTkSK$_{\text{road}}$) query [99]	Extensions of standard SK queries	RTkSK query	This query is based on the TkSK$_{\text{road}}$ query. It finds the set of objects that include a query object in their TkSK$_{\text{road}}$ query results.
Reverse top-k geo-social keyword (RkGSK) query [168]	Socially-aware queries	n.a.	This query is based on the TkGSK$_{\text{road}}$ query that extends a socially-aware SK query [13] from Euclidean space to consider road networks. It retrieves a set of users such that the TkGSK$_{\text{road}}$ query result of each returned user includes a given object.
Why-not top-k geo-social keyword (WNGSK) query [167]	Socially-aware queries	n.a.	Returns a minimally refined TkGSK$_{\text{road}}$ query that contains a set of missing objects in its result.
Collective SK (CSK) query [60, 172]	Group queries	MAX+MAX GSK query	This query finds a group of objects that are close to a query location, that are near each other and that collectively cover given query keywords.
Spatial group keyword search (SGKS) query [100, 101]	Group queries	n.a.	This query retrieves a set of objects, each of which has a set of neighbor objects that cover the query keywords collectively. An object's neighbor objects must be within a distance threshold of the object.
Group-based collective keyword (GBCK) query [132]	Group queries	n.a.	This query retrieves a region such that the objects in the region cover given query keywords. Further, the returned region has the minimum cost that considers the diameter of the region and the distances between the objects and the user locations.
Diversified SK (DSK) query [159]	Group queries	n.a.	Finds a group of objects that are both spatially and textually relevant to the query and that are diverse.
Keyword-aware route query [21, 78–80, 87, 90, 127, 154, 173]	Route planning	n.a.	Finds a route to a target location that optimizes a cost function that considers spatial and textual aspects of the objects on the route.

Table 4 Summary of the SK Queries in Road Networks (SK Denotes Spatial Keyword)

moving spatial attribute and a set of keywords as parameters. It continuously maintains an up-to-date list of k geo-textual objects (e.g., POIs) over the road network. Two types of moving spatial keyword queries exist: the moving Boolean kNN and the moving top-k spatial keyword query.

Moving Boolean kNN spatial keyword (MBkSK$_{road}$) query [175]: The MBkSK$_{\text{road}}$ query $q = (l_q, \psi, k)$ takes three parameters: l_q denotes a moving location, ψ denotes a set of keywords, and k is a positive integer. Each vertex of G has a set of keywords and can be considered as a geo-textual object. Let $S(q)$ denote the set of vertices in G such that $\forall v \in S(q)$, v's textual descriptions covers ψ. Then the result of q, $S_k(q)$ consists of k vertices in $S(q)$ such that $\forall v \in S_k(q)$ $(\forall v' \in S(q) \backslash S_k(q)$ $(dist(l_q, v) \leq dist(l_q, v')))$.

Moving top-k spatial keyword (MTkSK$_{road}$) query [62, 150]: This query extends the TkSK query to road networks straightforwardly.

Reverse spatial keyword (RTkSK$_{\text{road}}$) query [99]. The RTkSK$_{\text{road}}$ query extends the RTkSK query to road networks. To avoid repetition, we again skip its problem definition.

3.2.3 Socially-Aware Queries

Socially-aware queries assume that geo-textual objects $o \in O$ and social network users U are given that are located in a road network G. Each object o is a tuple (loc, key), where $o.loc$ is a location in G and $o.key$ is a set of keywords. Each user $u \in U$ is a tuple $(F(u), CI)$, where $F(u)$ is the set of friends of u in G, and CI is a set of check-ins of the form $CI_{u \to o_i}$ that captures the frequency at which u checks in at o_i.

Reverse top-k geo-social keyword (RkGSK) query [168]. This query takes as parameters an object $o_q \in O$ and an integer k. It retrieves a subset of users $U_q \subseteq U$ such that $\forall u \in U_q$, o_q is in the result of the TkGSK$_{\text{road}}$ query $q = (u, u.loc, u.key, k)$. The TkGSK$_{\text{road}}$ query extends a socially-aware spatial keyword query [13] in Euclidean space studies to consider road networks. Formally, given an object $o_q \in O$ and an integer k, the RkGSK query retrieve a subset of users $U_q \subseteq U$ such that $\forall u \in U_q, o_q \in S(u)$, where $S(u)$ denotes the result of the TkGSK query $q = (u, u.loc, u.key, k)$.

Why-not top-k geo-social keyword (WNGSK) query [167]. This query takes a TkGSK$_{\text{road}}$ query and a set of missing objects as parameters. It returns a minimally refined TkGSK$_{\text{road}}$ query that contains the missing objects in its result. Formally, given a TkGSK query $q = (u, loc, key, F, k)$ and a set of missing objects M, the WNGSK query returns a $TkGSKquery$ $q' = (u, loc, key', F', k')$ with a minimal penalty as computed by Eq. 13 and that includes M in its result. In the original query q, F is the set of social network friends of u, i.e., $F(u)$. The refined query q' may recommend a new set F' of social network friends of u.

$$P(q, q') = \lambda_1 \cdot \frac{max(0, R(q', M) - k)}{R(q, M) - k} + \lambda_2 \cdot \frac{Edit(key, key')}{MaxEdit(key, \cup_{o \in M} o.key)} + \lambda_3 \cdot \frac{|F' - F|}{H_u} \tag{13}$$

The penalty is composed of three weighted terms that capture the normalized difference of q' from the original query q. The weights sum to 1. The first term concerns the enlargement of parameter k, where $R(q, M)$ (resp. $R(q', M)$) denotes the maximal ranking of objects among M by query q (resp. q'). The second term captures the degree of change in the keywords, where $Edit(key, key')$ denotes the number of edits required to transform key into key', and $MaxEdit(key, \cup_{o \in M} o.key)$ denotes the maximum number of edits required to transform key into $o.key$ ($\forall o \in M$). The last term concerns the change in the social network friends of u, where $|F' - F|$ is the number of new friends recommended in the refined query, and H_u is the maximum number of friends that the social network system can provide to u.

3.2.4 Group Queries

A group query returns a group of geo-textual objects that collectively answer the query. Inter-object relations are considered for this type of queries. These queries rely on functions that capture the relevance of a group of objects, usually linear combinations of spatial proximity, textual relevance, and other aspects like the diameter group and the diversity of the objects in the group.

Collective spatial keyword (CSK) query [60, 172]. A CSK [60] query extends the MAX+MAX GSK query to road networks straightforwardly.

Zhao et al. [172] study a popularity-aware CSK query that assumes that geo-texutal objects have rating. The popularity-aware CSK query evaluates the goodness of a group of objects based on the sum of the objects' ratings while ensuring that the result objects are close to the query location and that the diameter of the group is within a threshold. Formally, a popularity-aware CSK query is given by $q = (\rho, \psi, \delta, \sigma)$, where ρ is a query location, ψ is a set of keywords, δ is a distance threshold, and σ is a diameter threshold. The result $S(q)$ of q satisfies that (i) $S(q) = \arg_S \max RS(S)$, where S is a group of objects and $RS(S)$ is the sum of the rating scores of the objects in S, (ii) $S(q)$ covers the keywords in ψ, and (iii) $\max\limits_{o \in S(q)} dist(o, q) \leq \delta \wedge \max\limits_{o_i, o_j \in S(q)} dist(o_i, o_j) \leq \sigma$.

Spatial group keyword search (SGKS) query [100, 101]. This query, given by $q = (r, \psi)$, where r is a distance and ψ is a set of keywords, is defined over a set of geo-textual objects (POIs) in a road network G. The result $S(q)$ of q is the set of POIs that satisfies that $\forall p \in S(q)(\exists O(\forall o \in O(dist(o, p) \leq r) \wedge \forall w \in \psi(\exists o \in O(w \in o.\psi))))$.

Group-based collective keyword (GBCK) query [132]. A GBCK query $q = (U, \psi)$ takes two parameters: U is a set of vertices of a road network, and ψ is a set of keywords. This query assumes that POIs with textual descriptions are located at road-network vertices. A group of users can issue a GBCK query to find a region that comprises a subset of vertices in the road network and that satisfies that (i) all query keywords are covered by the textual descriptions of the POIs located at the vertices, (ii) the user group is close to the vertices, and (iii) the vertices are close to each other.

The distance *Dist* between two vertices is the length of the shortest path between them. For a set of user vertices $U \subset V$ and a region R, the distance cost between U and R is computed as

$$Dist(U,R) = \max\{Dist(u_i, v_j) | u_i \in U, v_j \in R\}.$$

The diameter of R is computed as

$$Dia(R) = \max\{Dist(v_i, v_j) | \forall v_i, v_j \in R, v_i \neq v_j\}.$$

The cost of R is computed as

$$cost(R) = \alpha \cdot Dist(U,R) + (1-\alpha) \cdot Dia(R), \alpha \in [0,1].$$

Formally, a GBCK query $q = (U, \psi)$ returns a region R such that $R = \text{argmin}_R cost(R)$ and $\forall w \in \psi$, $\exists v \in R$, vertex v contains a POI that contains w in its textual description.

Diversified spatial keyword (DSK) query [159]. This query aims to find a group of objects taking into account the relevance and the spatial diversity of the objects. Formally, a DSK query is given by $q = (r, \psi, k)$ and takes as argument a set of objects located in a road network G. Parameter r is a point in G, ψ is a set of keywords, and k is a positive integer. The query returns a group of objects S such that (i) $|S| = k$, (ii) $\forall o \in S$, o contains all keywords in ψ, and (iii) for any other group S' of cardinality k, $f(S) \geq f(S')$, where function $f(\cdot)$ is defined as follows:

$$f(S) = \lambda \cdot Rel(S) + (1-\lambda) \cdot Div(S),$$

where $\lambda \in [0,1]$. Function *Rel* computes the relevance of S and is defined as follows:

$$Rel(S) = \frac{\sum_{o \in S}(1 - \frac{dist(o,q)}{dist_{max}})}{k},$$

where $dist(o,q)$ denotes the network distance between o and q and $dist_{max}$ denotes the maximum network distance. Finally, function *Div* computes the diversity of S and is defined as follows:

$$Div(S) = \frac{\sum_{o_i,o_j \in S} dist(o_i,o_j)}{k(k-1)\, dist_{max}}.$$

3.2.5 Route Planning

Keyword-aware route queries [21, 78–80, 87, 90, 127, 154, 173]. This type of query assumes a set of objects (POIs) that are located in a road network G. A query is given by $q = (v_s, v_t, \Psi, \Delta, f)$. Here, v_s and v_t are a source and a target location in G. Next, Ψ is a *visit order graph*, a directed acyclic graph where vertices correspond to keywords and an edge (a,b) denotes that POI with the keyword of a must be visited before a POI with the keyword of b. Keyword matching can be either exact or approximate. Approximate matching uses a string similarity (e.g., edit distance) threshold to determine whether a term in an object matches a term in a query. Zheng et al. [173] study a matching strategy based on so-called clues. Specifically, a clue is defined as $\mu(w,d,\epsilon)$, where w is a keyword, d is a network distance, and $\epsilon \in [0,1]$ is a confidence value. A road segment (u,v) matches a clue μ if vertex v contains w and the network distance between u and v is within $[d(1-\epsilon), d(1+\epsilon)]$. Next, Δ is an optional budget limit (e.g., travel distance threshold). Finally, f is a function that calculates a score of a route, e.g., route popularity. The query returns a path R in G that starts at v_s and ends at v_t optimizes $f(R)$ while satisfying the budget limit Δ and passing through locations in a sequence given by Ψ.

Example: Find the most popular route from and to my hotel (v_s, v_t) that first passes a *shopping mall* and a *restaurant* and then a *pub* (ψ) and such that the time spent on the road is within 4 hours (Δ).

Table 5 categorizes the route queries based on keyword covering types, object score, and budget score. Note that *order* under *Keyword Covering* denotes whether the query keywords are must be covered in a one-after-another fashion by the POIs.

4 Querying Static Geo-textual Data - Methodology

4.1 Indexing

The efficient processing of spatial keyword queries over static geo-textual data typically relies on the effective indexing of the geo-textual data and on search algorithms that exploit available indices.

We present the existing indexing structures for geo-textual data. They are based on an existing spatial index and an existing textual index and combine the two types of indices in different ways.

4.1.1 Spatial Indexing

We classify spatial indices into three categories, namely R-tree-based indices, grid-based indices, and space-filling-curve-based indices.

R-tree-based. Three indices use the R-tree [66]. Most geo-textual indices belong to this category and use the inverted file for text indexing. In early work [177], the R-tree-based indices combine the R-tree and inverted files loosely to organize the spatial and text data sep-

Literature	**Keyword Covering**			**Objective score**	**Budget score**
	order	word match	word weight		
Sharifzadeh et al. [127]	ordered	exact	none	distance	none
Zheng [173]	ordered	exact	none	distance	none
Li et al. [87]	partially ordered	exact	none	distance	none
Cao et al. [21]	not ordered	exact	none	non-distance	distance
Li et al. [90]	not ordered	exact	none	non-distance	distance
Yao et al. [154]	not ordered	approximate	none	distance	none
Kanza [78]	not ordered	exact	none	distance	none
Kanza [79]	not ordered	exact	none	distance	distance
Kanza [80]	not ordered	exact	yes	text relevance, distance	distance

Table 5 Overview of Keyword-aware Route Queries

Index	**Spatial part**	**Textual part**	**Combination Scheme**	**Targeted Query**
bR*-tree [161]	R*-tree	signature	loosely spatial-first	mCK
CIR-tree [48]	R-tree	inverted file	tightly spatial-first	TkSK
CIR$^+$-tree [28]	R-tree	inverted file	tightly spatial first	Time-aware BSK
IF-R*-tree [177]	R*-tree	inverted file	loosely text-first	BRSK
I^3 [163]	Quad-tree	inverted file	loosely text-first	BRSK, BkSK, TkSK
ILQ (disk part) [158]	Quad-tree	inverted file	loosely text-first	BkSK, TkSK
ILQ (memory part) [158]	Quad-tree	signature	loosely spatial-first	BkSK, TkSK
Inverted KD-tree [86]	Kd-tree	inverted file	loosely spatial-first	skyline
Inverted R-tree [61]	R-tree	inverted file	loosely text-first	BRSK
IR-tree [48, 143]	R-tree	inverted file	tightly spatial-first	TkSK
IR-tree [91]	R-tree	inverted file	tightly spatial-first	TkSK
IR^2-tree [76]	R-tree	signature	tightly spatial-first	BkSK
IUR-tree [95, 97]	R-tree	intersection/union vector	tightly spatial-first	RTkSK
KR*-tree [68]	R*-tree	inverted file	tightly spatial-first	BRSK
LIR-tree [29]	R-tree	inverted file	tightly spatial-first	TkSK
RSR-tree [92]	R-tree	signature	tightly spatial-first	clue-based query
R*-tree-IF [177]	R*-tree	inverted file	loosely spatial-first	BRSK
S2I [122]	R-tree	inverted file	loosely text-first	TkSK
SF2I [41]	SFC	inverted file	loosely spatial-first	BRSK
SFC-Q [45]	SFC	inverted file	*key_func*(spatial, text)	BRSK
SI [133]	R-tree	inverted file	loosely text-first	BkSK
SIS [55]	R-tree	inverted file	loosely text-first	RTkSK
SKI [25]	R-tree	signature	loosely spatial-first	BkSK
SKIF [81]	Grid	inverted file	*key_func*(spatial, text)	BRSK
ST [135]	Grid	inverted file	loosely spatial-first	BRSK
ST2I [71]	kd-tree	signature	tightly spatial-first	BRSK, TkSK
STbHI [102]	SFC	inverted file	*key_func*(spatial, text)	BRSK
TS [135]	Grid	inverted file	loosely text-first	BRSK
Virtual bR*-tree [162]	R*-tree	signature	loosely spatial-first	mCK
WIBR-tree [146]	R-tree	inverted bitmaps	tightly spatial-first	BkSK

Table 6 Comparison of Hybrid Geo-Textual Indices

arately. Subsequent indices integrate the R-tree with a text index tightly (e.g., [48]).

Grid-based. These indices combine a grid-based index with a text index (e.g., the inverted file). The grid-based indices divide space into a predefined number of square or rectangular equal-sized cells. The grid index and the text index can be either separate [135] or combined tightly [81].

Space-filling-curve-based. These indices combine inverted files with a space filing curve, and they include a Hilbert-curve-based index [41] and a Z-curve-based index [45]. These indices rely on the property that the points close to each other in the native space are also close to each other on the space-filling curve.

Quad-tree-based. Two geo-textual indices, I^3 [163] and ILQ [158], use the Quad-tree [125] as the spatial index. The Quad-tree is a spatial index that recursively divides the spatial region into four congruent rectangular cells.

4.1.2 Text Indexing

Inverted file. Most geo-textual indices [45, 48, 68, 81, 91, 122, 135, 177] use the inverted file for text indexing.

An inverted file has a vocabulary of terms, and each term is associated with an inverted list that comprises a sequence of postings, each of which normally contains the identifier of an object o, whose description $o.\psi$ contains the term, along with the frequency of the term in $o.\psi$. The frequency information is not included in indices targeting Boolean queries. In general, the postings in each inverted list are sorted by object ID. However, some geo-textual indices order the postings differently, such as ordering them by their orders in grid cells [81] or according to a space-filling curve [45].

Signature/Bitmap. Some R-tree-based indices [25, 76, 146] use a signature file [53] to index the text information in subtrees. A Bitmap is the simplest case of a signature. Simply put, each bit in a signature represents the presence or absence of a term in a document. The IR^2-tree [76] augments each node of the R-tree with a signature file to capture the text information of the objects in the node's subtree. Another study [161] augments the nodes of the R-tree with bitmaps. Some geo-textual indices [146] use inverted bitmaps, in which each term corresponds to a bitmap and each bit in a bitmap captures whether a document contains the term. In addition, ILQ [158] maintains a signature Quad-tree in memory. Here, each cell stores a signature that indicates the existence of each keyword in the cell.

4.1.3 Combination Schemes

Hybrid indexing schemes combine spatial and text indexing. We categorize the indices according to how they combine the two, e.g., text-first combination or spatial-first combination. Table 6 summarizes existing hybrid indices based on their spatial indexing schemes, their text indexing schemes, and ways of combining spatial and text indexing.

A text-first combination index usually employs the inverted file as the top-level index and then applies a spatial index to the postings in each inverted list, which can be an R-tree, a grid, or a spatial-filling curve. In contrast, the top level of a spatial-first index is a spatial structure, and its nodes (resp. grid cells) contain inverted files or bitmaps for the text information of objects contained in the nodes (resp. grid cells).

Some combinations are loose in a way that the data is indexed by one type of index and then by the other type of index, thus pruning the search space one after the other during query processing. In contrast, tight integrations enable the use of both types of information for pruning the search space during query processing (e.g., [45, 48]). More details about indices for geo-textual data can be found in an experimental evaluation study [32].

4.2 Querying over Euclidean Space

4.2.1 Standard Queries

An experimental evaluation study [32] already provides a good coverage of algorithms for standard spatial keyword queries. Here, we only give a high-level overview. In loosely combined spatial-first indices, during the query processing, the geo-textual objects are first filtered based on the spatial index, and when the leaf level is reached, the textual index is accessed to obtain the objects satisfying the query. The spatial filtering stage may generate a large number of candidate objects.

In loosely combined text-first indices, the text index is first used to find the spatial indices relevant to the query keywords, and the spatial indices are then used to find the objects that are spatially relevant to the query. The problem of this method is that an object is contained in all spatial indices corresponding to its keywords. Thus we may need to access multiple spatial indices for the same geo-textual objects.

In tightly combined indices (i.e., tightly spatial-first and *key_func*(spatial, text) in Table 6), the BRSK query processing algorithms usually adopt the *filtering-and-refinement* paradigm. For example, for the KR*-tree, the search algorithm first finds the R*-tree nodes that contain the query keywords and then uses these as candidates for subsequent search. This approach suffers from unnecessary overhead when there are many candidates. Similarly, the SFC-Q algorithm first traverses the Quad-tree to obtain the object ID ranges that contain all objects intersecting the query range. Then the inverted lists are swept to fetch the needed parts. The search algorithms for tightly combined indices for the BkSK and TkSK queries utilize *best-first* traversal. In best-first traversal, a priority queue is used to keep track of the nodes and objects that have yet to be visited. The priority of each node is an upper bound (or lower bound, depending on the scoring function) of the ranking score in terms of the query and the objects in the node.

In addition, a proposal [83] presents a query plan optimizer to handle BRSK queries using a spatial index and a keyword index.

4.2.2 Standard Query Extensions

Temporal spatial keyword query [28, 113, 116]. Nepomnyachiy et al. [116] and Mehta et al. [113] investigate the TBRSK query and variants of it. They store objects in "end" nodes of a spatial index (the leaf nodes of an R-tree or cells of a Grid index). The difference is how they organize the objects in an end node. Nepomnyachiy et al. first use an inverted index to categorize the objects, sorting the objects in inverted lists in ascending order of their timestamps. In contrast, Mehta et al. use a KR*-tree to organize the objects. Mehta et al. consider moving data and split the data into multiple objects each with a timespan and a set of keywords. The KR*-tree organizes the objects based on their timespans (each timespan can be considered as a line in Euclidean space) and keywords.

Chen et al. [28] study the time-aware Boolean spatial keyword query, which finds top-k objects using a ranking score considering the objects' temporal and spatial attributes. They use a CIR$^+$-tree to organize the objects. When processing a query, they traverse the CIR$^+$-tree. The algorithm starts by putting the root node into a max-heap based on the ranking function. In each step, it fetches a node from the max-heap. If a non-leaf node is obtained, its child nodes that contain the query keywords are put into the max-heap. A child node is ignored if it does not have any object that may appear in the top-k result. When obtaining a leaf node, the algorithm computes the ranking scores of the contained objects and updates the top-k result accordingly. The algorithm terminates when the max-heap is empty.

Socially-aware spatial keyword query [13, 77, 145]. Ahuja et al. [13] and Wu et al. [145] employ a tree index to organize the objects. The objects are stored in the leaf nodes, and each intermediate node maintains summary information on the objects in the leaf nodes of the node's subtree. The summary information is used for computing a bound on the scores of the objects in the corresponding subtree, thus enabling filtering of irrelevant objects without reaching the leaf nodes. Ahuja et al. [13] use an index that is similar to a Quad-tree, and Wu et al. [145] extend the IR-tree.

Jiang et al. [77] categorize the objects using a distributed spatial textual index and store the objects in an HDFS [4]. They first organize the objects using a Quad-tree and then compute a Geohash code of each leaf node by using its space-filling-curve value. Each pair of a Geohash code and a keyword has a posting list of objects that are stored in HDFS. To answer a query, they first compute the set of relevant Geohash code and keyword pairs and then process the objects in the corresponding posting lists.

Direction-aware spatial keyword query [84]. Li et al. [84] propose a direction-aware index to organize objects. Using a tree structure, the objects in the non-leaf nodes are partitioned into sub-regions according to their directions and distances with respect to the bottom-left point of a Minimum Bounding Rectangle (MBR) that covers them. When processing a query, the index allows to prune the nodes whose objects are outside the query direction interval, thus enabling good filtering.

Preference-aware spatial keyword query [15, 88, 134]. Existing studies [15, 88, 134] focus on developing indexing schemes to store feature objects and, if any, their corresponding preference scores. The proposed indices are based on the R-tree and inverted file.

Top-k prestige-based spatial keyword (TkPSK) query [23]. Cao et al. [23] propose an exact and an approximation algorithm using an IR-tree. The high-level idea of the exact algorithm is to consider only nearby IR-tree nodes when propagating prestige-based relevance (PR), which speeds up computing the PR score substantially. The approximation algorithm groups objects into subgraphs based on their locations. Each subgraph corresponds to a leaf node of the IR-tree. The main idea is based on the observation that the PR scores of the nodes in a subgraph can be computed by PR propagation within the subgraph and contributions from border objects that connect the subgraph with other subgraphs. Consequently, the computation of PR scores can be done on subgraphs rather than having to be done on the full graph.

Moving spatial keyword query [74, 142, 147, 148]. This query has a moving location and a set of keywords. The high-level idea of existing work is based on the concept of *safe regions*. In particular, a safe region of a query is a region that includes the query location. If a moving query remains in the safe region, its result will not change, meaning that it is not necessary to request a new result from the server. The client monitors whether the query location is inside its safe region. If not, the client sends a request to the server. The server computes a new result and a new safe region, and sends them to the client.

Reverse spatial keyword queries [55, 95–97, 171]. Considering the RBkSK query, two studies [55, 171] propose algorithms that use an inverted index and an R-tree separately. They first use the inverted index to find the candidate objects that contain the query keywords, then use the R-tree to compute the degree of influence of those candidates on the spatial dimension. To accelerate the procedure, they use half-planes to prune the invalid objects and compute the influence on the query.

Considering the RTkSK query, Lu et al. [95–97] propose an *Intersection-Union-R tree* and two variants (i.e., IUR-tree, CIUR-tree, and C^{2}IUR-tree) to compute the query efficiently. The IUR-tree extends the IR-tree as follows: Each non-leaf node additionally maintains an entry that points to an intersection textual vector and a union textual vector where each item (dimension) corresponds to a distinct term that appears in the objects stored in the node's subtree. The two vectors enable computing the lower and upper bounds of the textual similarity. The variants of the IUR-tree (i.e., CIUR-tree, and C^2IUR-tree) enrich the entry by adding textual cluster information. This information is used to filter irrelevant nodes when traversing the tree.

Spatial keyword skyline query [86,121,128]. Some studies [86, 121] propose hybrid index structures to organize objects and develop corresponding pruning techniques. Li et al. [86] propose an *Inverted KD-tree*, which is a KD-tree where each leaf node is extended with inverted files. Regalado et al. [121] use an IR-tree. Both studies employ a search algorithm that originates from the Block Nested Loops (BNL) scheme [18], which is a popular method for supporting the skyline operator in relational databases.

Shi et al. [128] use an R-tree to organize objects. To speed up computing domination relations, they exploit geometric properties of the problem to filter query points that have no impact on the inclusion/exclusion of any object in/from the skyline. They develop three models for answering different spatial keyword skyline queries, summarized as follows:

(1) *Derived Dimension Augmentation (DDA)* adds textual relevance to the dimensions of spatial skylines.

(2) *Keyword Boolean Filtering First (KBFF)* is a two-step processing algorithm. It first selects candidate objects whose textual information contains at least one of the query keywords and then computes the spatial skyline of the candidates.

(3) *Spatio-Textual Dominance (STD)* converts the spatial distance measure of the spatial skyline dimensions into a combined spatio-textual relevance measure; hence, skylines can be computed that both spatial and textual relevance are taken into consideration.

4.2.3 Group Queries

Group spatial keyword queries [22, 24, 26, 27, 43, 50, 64, 94, 110, 130, 161, 162, 164]. Based on their optimization goals, existing studies can be classified into two categories. The first category of studies focus on optimizing a cost function that considers inter-object distance and the distance between objects and the query location. The other category of studies consider a cost function involving aspects beyond distance. We proceed to consider each category in turn:

We observe that it is NP-hard to find an exact answer to queries in the first category [22, 64]. Table 7 gives a brief overview of existing studies.

For the mCK query, Zhang et al. [161] develop an exact algorithm based on the bR*-tree that utilize two monotone constraints, distance mutex and keyword mutex. The distance mutex is based on the observation that if the distance between the MBRs of two nodes exceeds a value θ then these two nodes cannot give a result with diameter better than θ. The keyword mutex has properties similar to the distance mutex. In subsequent work [162], the authors propose an improved version of the bR*-tree, the virtual bR*-tree, which improves the query efficiency. Guo et al. [64] develop three approximation algorithms for the mCK query that exploit the virtual bR*-tree. The first algorithm is a greedy approach that has an approximation ratio of 2. The other two algorithms find the circle with the smallest diameter that encloses a set of objects that cover all query keywords collectively, called the "smallest keywords enclosing circle (SKEC)." They prove that SKECs enable answering the mCK query with an approximation ratio of $\frac{2}{\sqrt{3}}$. Thus, the query can be answered with an approximation ratio of $\frac{2}{\sqrt{3}} + \epsilon$ (ϵ is an arbitrarily small value). In addition, Guo et al. [64] develop an exact algorthm based on SKEC.

For the SUM-GSK query, Cao et al. [24] propose an approximation algorithm and an exact algorithm. The former algorithm traverses the IR-tree while maintaining a min-priority queue. The cost of a node is computed by dividing the minimum distance between the node and the query by the number of their common keywords. The exact algorithm is based on dynamic programming. An additional dynamic programming algorithm is proposed in subsequent work [22]. He et al. [69] propose a distributed, exact algorithm for the SUM-GSK query. They partition the objects according to a grid so that they can be processed in parallel. For each partition, a local optimal result is computed, and then these results are aggregated to obtain the final result.

For the MAX+MAX GSK query, Cao et al. [24] propose one exact algorithm and two approximation algorithms, denoted by $\mathbf{Approx}_1$ and $\mathbf{Approx}_2$. $\mathbf{Approx}_1$ finds the nearest object for each query keyword and builds a group consisting of the objects found. $\mathbf{Approx}_2$ improves $\mathbf{Approx}_1$ by utilizing the least frequent query keyword (denoted by t_{inf}). It creates a new query using t_{inf} and calls $\mathbf{Approx}_1$ to obtain a result R_1. Then it calls $\mathbf{Approx}_1$ for the original query to obtain a result R_2. It outputs the result with the smaller cost. The exact algorithm is equipped with a set of prunning strategies designed for an exhaustive search for each object to find the optimal group containing t_{inf}. In subsequent work [22], a new approximation algorithm for the MAX+MAX GSK query is proposed, and a new exact algorithm is also proposed that per-

Literature	Target Query	Index	Approach Description
Zhang et al. [161]	mCK	bR*-tree	An exact algorithm that utilizes two monotone constraints, distance mutex and keyword mutex, to prune irrelevant nodes when traversing a bR*-tree.
Zhang et al. [162]	mCK	virtual bR*-tree	A virtual bR*-tree index and associated query processing techniques.
Guo et al. [64]	mCK	virtual bR*-tree	Three approximation algorithms, SKEC, SKECa, and SKECa+, and an exact algorithm based on SKECa+.
Cao et al. [22,24]	SUM-GSK	IR-tree	An approximation algorithm that divides the distance between a node and a query by their common keywords, and an exact algorithm.
He et al. [69]	SUM-GSK	Grid	A distributed, exact algorithm that parallelizes the checking of objects in different grid cells.
Cao et al. [22,24]	MAX+MAX GSK	IR-tree	An exact algorithm and two approximation algorithms that partition the objects into groups based on the query keywords.
Long et al. [94]	MAX+MAX GSK	IR-tree	An exact algorithm and an approximation algorithm.
Cao et al. [22]	MIN+MAX GSK	IR-tree	An approximation algorithm.
Chan et al. [27]	generalized GSK	n.a.	A unified cost function capturing different types of GSK queries and an exact algorithm and an approximation algorithm for the unified cost function.

Table 7 Studies on Group Queries that Consider Distance-based Cost Functions

forms better than the earlier exact algorithms [24,94]. Long et al. [94] propose an exact algorithm and an approximation algorithm for the MAX+MAX GSK query. The exact algorithm is based on the observation that the maximum cost of a group is dominated by at most three objects: the object with the largest distance to the query and the two objects having the largest pairwise distance. The approximation algorithm achieves a smaller approximation factor than do existing algorithms [22, 24] with higher complexity. It recursively searches the o-neighborhood feasible set for the objects relevant to the query. The o-neighborhood feasible set of an object o is the set containing o and all other objects, each of which is the t-keyword nearest neighbor of o in $R(q, o)$ for each $t \in q.\psi \setminus o.\psi$, where $R(q, o)$ is the circle centered at query location q that has a radius equal to the distance between q and o. The t-keyword nearest neighbor of o is the object nearest o that contains keyword t.

For the MIN+MAX GSK query, Cao et al. [22] propose an approximation algorithm similar to $\mathbf{Approx}_1$ for the MAX+MAX GSK query. The difference concerns how the cost of a group of objects is computed.

Chan et al. [27] consider the generalized GSK query. They propose a unified approach that supports a unified cost function that can be instantiated to existing cost functions. Their approach consists of an exact algorithm and an approximation algorithm. The approximation algorithm provides a better approximation than do existing solutions.

We proceed to consider the category of studies that consider a cost function involving aspects beyond the distance. Chan et al. [26] propose an exact algorithm and an approximation algorithm for the inherent-cost aware GSK query. The exact algorithm accesses objects relevant to the query in ascending order of their distance to the query. For each relevant object o, the algorithm constructs the best feasible group using o as the object contributing the query-object distance in the cost function. The approximation algorithm is similar, except that in each iteration, it constructs a feasible group greedily rather than exhaustively.

Zhang et al. [164] propose an exact algorithm and an approximation algorithm for the level-aware GSK query. The former employs a keyword hash table that maintains the mapping from keywords to lists of objects. It recursively accesses the relevant objects and constructs the feasible group until it finds the optimal feasible group. The approximation algorithm conducts search on an IR-tree where the nodes are extended to include level and cost information.

Clue-based spatial keyword search [92]. Liu et al. [92] propose solutions to clue-based spatial keyword search. A clue is specified in terms of categories of objects near the target object. They present a roll-out-star R-tree (RSR-tree) index that extends the nodes of an R-tree with spatio-textual context information. The query algorithm performs a best-first traversal of the RSR-tree and computes the score of a node using the node's context information. The object with the largest score is the result.

Spatial pattern matching [56]. A spatial pattern P is a graph where each vertex represent a geo-textual object and each edge represents the distance between two objects. An $\underline{\text{m}}$ulti-$\underline{\text{p}}$air-$\underline{\text{j}}$oin (MPJ) algorithm is proposed that finds matches for the edges of P in order. To decide the execution order, a sampling-based method is used to estimate the processing costs of different execution orders. An IR-tree on the collection of objects is employed to accelerate finding edge matches. The found matches are linked incrementally to form subgraphs, which are then output as results. Further, a multi-star-join (MSJ) algorithm is proposed that improves the efficiency by improved pruning.

Top-k spatial textual clusters query [130, 144]. Existing studies differ in how they find the candidate objects that compose a cluster. Skovsgaard and Jensen [130] consider objects that are textually relevant to the query based on a similarity function. In contrast, Wu and Jensen [144] only consider objects that contain at least one query keyword. Skovsgaard and Jensen [130] employ a group extended R-tree (GER-tree) in which each non-leaf node maintains a compressed histogram containing summary information on its subtree. As in other top-k algorithms, they maintain a priority queue while traversing the GER-tree. In each iteration, they compute a bound on nodes to facilitate filtering. Wu and Jensen [144] solve the query in a different way. They use an IR-tree to find the objects that contain at least one query keyword. They then construct clusters on those objects using DBSCAN [119]. They also introduce so-called spatially gridded posting lists to prune sparse neighborhoods while forming clusters.

4.2.4 (Top-k) Spatio-textual Joins

Most studies [19, 54, 75, 93, 120] focus on the use of indexing in order to efficiently find object pairs that are spatially and textually similar. Liu et al. [54, 93] design a spatial signature and a textual signature for each object that are then used to prune dissimilar object pairs. Rao et al. [120] develop two spatial-first and two text-first indexing schemes. Hu et al. [75] generate a spatio-textual signature set for each object and leverage these sets to prune dissimilar object pairs. Bouros et al. [19] propose different spatial-index-based algorithms. They also propose a batch processing mechanism that partitions the objects into groups based on their spatial and textual attributes and then performs joins on the groups. Zhang et al. [166] propose a MapReduce framework to solve the join problem.

4.2.5 Region Finding/Analysis

Geosocial search [117]. Geosocial search [117] finds geographical regions based on geo-tagged social network posts in the regions. Three models are used to quantify the relevance of a region to a query: Global Ratio, Local Ratio, and Harmonic Mean. Geosocial search is performed using a partition-aware inverted index on the geo-tagged posts. This index partitions the space using a grid and maintains an inverted index on the posts in each grid cell. The relevance score of a cell to a query is the number of relevant posts in the cell. Geosocial search finds k cells with the largest relevance scores and merges adjacent cells into polygon for visualization.

Reverse top-k keyword-based location (Reverse_TkSK) query [149]. Xie et al. [149] propose to use Voronoi cells to represent spatial regions. Given a set O of spatial points, the Voronoi cell of an object $o \in O$ is the part of the space that contains all points having o as their nearest neighbor. To compute the result of a Reverse_TkSK query, i.e., a spatial region, a Quad-tree is used for approximating the result Voronoi cells V_q. During the construction of the Quad-tree, each Quad-tree cell is furthered partitioned depending on its relation with V_q. An IR-tree on the objects is employed to accelerate the checking.

Top-k most frequent terms query [12]. Ahmed et al. [12] use an R-tree-based index to compute the top-k most frequent terms. In particular, four indices are presented, which are called STL-L, STL-LI, STL-Li, and STL-li. They all augment some nodes with sorted term lists (STLs). A node's STL contains aggregated term entries based on the objects in the node's MBR. A term's entry stores the term's frequency and information on the objects in which it occurs. Each STL is sorted descendingly on the term frequency. After accessing candidate nodes, two popular top-k algorithms, Random Access (RA) [52] and Non Random Access (NRA) [115], are employed to compute the result. The STL-L index augments the leaf nodes with STLs, while STL-LI augments both non-leaf nodes and leaf nodes with STLs. STL-LI allows early termination when accessing some non-leaf nodes, at the cost of increased memory consumption. STL-Li reduces the memory consumption by reducing the lengths of STLs in both non-leaf nodes and leaf nodes.

Topic exploration [169, 170]. Zhao et al [169, 170] organize the geo-textual data using an Octree [112]. Unlike in the conventional Octree, they present an algorithm to determine cells in which to pre-train topic models such that the memory consumption does not exceed a threshold and the error rate is below a bound. To support efficient exploration of topics at different granularities, they propose efficient means of combining topic models of different cells.

4.2.6 Query Modification

Why-not spatial keyword queries [33, 34, 38, 39, 176]. Chen et al. [33, 34, 38, 39] propose solutions to different why-not queries. To facilitate α and k modification, they transform the candidate parameter vectors into 2-dimensional vectors and classify vectors as promoted points or degraded points. They search only candidate vectors through promoted points to avoid exhaustive enumeration. They also propose a Bounded IR-tree to prune unnecessary accesses to objects and promoted points. To support direction modification, they develop a linear programming algorithm. To enable adapting the query keywords, they propose an algorithm that performs pruning using a Keyword count R-tree, which is an R-tree that augments each node with textual information on the indexed objects. Zheng et al. [176] propose a 3-phase solution for interactive, preference-aware TkSK query modification.

Why-not group spatial keyword query [174]. Zheng et al. [174] propose an approximate solution to the why-not group spatial keyword query. They employ an IR^2-tree to retrieve objects that need to be considered when refining the query. They propose an incremental sampling approach to select good weight vectors using three heuristic strategies: a score based strategy, a weight modification strategy, and a rank improvement strategy. They find the vector with the lowest penalty among the selected weight vectors.

4.3 Querying over Road Networks

4.3.1 Standard Spatial Keyword Queries on Road Networks

Boolean range spatial keyword (BRSK) query [101]. Luo et al. [101] propose distributed means of computing the BRSK query. They partition the road network into N subgraphs, i.e., N partitions, each of which is assigned to a virtual machine (VM). The partitioning information is stored in a component called the partitioner. For each partition P, an NPD-index is built that maintains information on the distance from any node in the road network to any node in P. To process a range keyword query, the partitioner first identifies the partitions covering the query range and keywords. Then the corresponding VMs compute partial results in parallel. Finally, the partial results are aggregated to obtain the query result.

Top-k kNN spatial keyword (TkSK) query [123]. Rocha-Junior and Norvåg [123] propose indexing that introduces a spatio-textual index (e.g., the IR-tree) into a road network framework. In the proposal, an inverted index maintains inverted lists with a key composed of an edge identifier and a term. In particular, the inverted list for an $(edge, term)$ pair stores the set of objects lying on $edge$ and having $term$ in their textual description. A B-tree like structure is used to map keys to their inverted lists to efficiently obtain the set of objects relevant to a given $(edge, term)$ pair. An algorithm similar to Dijkstra's algorithm [51] uses the indexing. To further improve efficiency, an overlay network on top of the road network is used for pruning regions that contain no result objects.

4.3.2 Extensions of Standard Spatial Keyword Queries

Moving Boolean kNN spatial keyword (MBkSK$_{\mathbf{road}}$) query [175]. Zheng et al. [175] preprocess a road network G to construct an index that augments the vertices in G with distance and keyword information. The index enables pruning of vertices that are far from the query location or contain no query keywords. To avoid frequent computations caused by the changing query location, they compute and maintain a path that is called a dominance interval with the property that the query result remains correct as long as the query location is in the interval. This reduces the computation and communication overhead significantly.

Moving top-k spatial keyword (MTkSK$_{\mathbf{road}}$) query [62]. Guo et al. [62] propose two algorithms for the MTkSK$_{\text{road}}$ query: a query-centric algorithm (QCA) and an object-centric algorithm (OCA). Both algorithms transform the problem into checking the vertices of the road network, which is achieved by traversing the road network. QCA starts from the end vertex where the query resides. It recursively visits the neighboring vertices until finding the top-k results. An expansion tree is maintained to prune nodes that contain no results. OCA adopts a different strategy. It first finds the relevant objects that contain the query keywords. Then it traverses the road network starting from a node where a relevant object resides. A shortest path tree is built when traversing the road network, which can be used to answer queries whose location resides in a node in the tree.

Reverse spatial keyword query [99]. Luo et al. [99] propose an algorithm that first traverses the road network to obtain a set of candidate objects O_c. A priority queue is maintained to store the unvisited edges in ascending order of their distance to the query location. For each object $o_c \in O_c$, the algorithm checks whether $o_q \in \mathrm{T}k\mathrm{SK}_{\mathrm{road}}(o_c)$, where o_q denotes the query object. During this procedure, if an object o_c has been ruled out, several unchecked objects in O_c can be pruned. To accelerate the checking, the algorithm exploits a Network Voronoi Diagram index that maintains the information about the distances between the objects in the road network.

4.3.3 Socially-Aware Query

Reverse top-k geo-social keyword (RkGSK) query [168]. Zhao et al. [168] propose an algorithm that splits the users and objects into groups and then for each group of users computes lower and upper bounds of the geo-social keyword similarity with each group of objects. These bounds are used for pruning. In the refinement phase, the remaining objects are used to compute the result. An index is also introduced to facilitate computing similarity values.

Why-not top-k geo-social keyword (WNGSK) query [167]. Zhao et al. [167] enumerate different parameters of the queries to generate a set of refined queries. Early termination techniques are employed to reduce the search space. To compute the penalties of refined queries, they build a PIM-tree, which integrates a Partitioned road network, an Inverted Intersection-Union file, and a Checkin-in&Friendship Matrix. They create a query partition tree (QP-tree) to partition the refined queries, and use it and the PIM-tree to prune non-result refined queries based on penalty bounds. Finally, a refined query with the minimum penalty is obtained.

4.3.4 Group Queries

Collective spatial keyword (CSK) queries [60, 172]. Gao et al. [60] prove that the CSK query is NP-complete and propose two approximation algorithms: a network-expansion-based (NEB) algorithm and an iterative-NEB-based (INB) algorithm. The NEB algorithm finds the edge that the query resides on and traverses the road network. For each distinct query keyword, the first found object that contains that keyword is added to the result. The INB algorithm improves the approximation bound of the NEB algorithm by considering both query-object distances and pairwise object distances.

Zhao et al. [172] solve the popularity-aware CSK query. They propose an exact algorithm that employs an index to facilitate finding the shortest paths between vertices. They also propose a heuristic algorithm. The road network is first partitioned into multiple subgraphs, and a multi-level index, called the I^3ndex, is built on them. The I^3ndex contains a local index for each subgraph and a global index. The algorithm works by traversing the subgraphs using the I^3ndex.

Spatial group keyword search (SGKS) query [100, 101]. Luo et al. [100, 101] propose a system composed of three components: a partitioner, an indexer, and a query processor. The partitioner partitions the road network into N subgraphs. Then the indexer builds an NPD-index on each partition that stores the distances between the vertices of the road network. The query processor is deployed on a pool of virtual machines. To process a query, each query processor instance first uses the NPD-index on its partition to compute the set of vertices satisfying the distance constraint for each query keyword and then intersects the resulting sets of vertices. Finally, the partial results are aggregated to obtain the result.

Group-based collective keyword (GBCK) query [132]. Su et al. [132] propose an exact algorithm and an approximate algorithm. Both algorithms initially find the first feasible region, which is achieved by checking the vertices nearest to the user vertex for each query keyword. The exact algorithm enumerates regions based on the first feasible region. It recursively selects the nearest unprocessed vertex and constructs a new feasible region. Finally, it outputs the feasible region having the minimum cost. The approximation algorithm recursively selects the nearest unprocessed vertex v to the user group and constructs a feasible v-centralized region. The algorithm terminates if no feasible v-centralized regions exist.

Diversified spatial keyword (DSK) query [159]. Zhang et al. [159] first employ incremental network expansion to retrieve candidate objects that satisfy the spatial and keyword constraints. Then, they propose a diversified spatial keyword search algorithm. The algorithm maintains a variable θ_T, which records the shortest diversification distance for the objects seen so far. For an object o, if no other candidate object o' satisfying $\theta(o, o') \geq \theta_T$ exists, o is pruned, where $\theta(o, o')$ denotes a function computing the diversification distance between o and o'.

4.3.5 Route Planning

Keyword-aware route queries [21, 78–80, 87, 90, 127, 154, 173]. The problem of answering keyword-aware route queries can be viewed as a generalized traveling salesman problem [21] and is NP-hard. An exception is the clue-based route search problem [173]. Most existing studies focus on developing heuristic algorithms, and some studies present both exact and heuristic algorithms. We classify existing solutions into three categories.

(1) *Exact algorithms* [90, 127, 173]: The exact algorithms enumerate all feasible routes to answer a keyword-aware route query. They employ different pruning techniques to reduce the computational costs. For example, Li et al. [90] start by finding the shortest path between the start and end vertices in the query and then recursively refine the route by adding vertices containing uncovered keywords. Sharifzadeh et al. [127] find the optimal route by enumerating the nearest neighbors to different point sets until reaching the start and end vertices in the query.

(1) *Heuristic algorithms without bounds* [78–80, 87, 90, 173]: A greedy algorithm is proposed that keeps selecting the next vertex in the road network greedily by taking into account an objective score, budget score (if any), and keyword coverage. This procedure is repeated until the target vertex is reached. The search order can be reversed: it is possible to start from the target vertex and conduct the greedy selection until the source vertex is reached. The algorithm is very efficient, but it does not offer a guaranteed approximation ratio.

(2) *Approximation algorithms* [21, 154]: Some studies provide approximation algorithms that have theoretical guarantees. Yao et al. [154] develop a global minimum path algorithm with an approximation ratio of κ, where κ is the number of query keywords. Cao et al. [21] scale the objective score of every edge of the road network by a parameter ϵ to obtain a scaled graph. They conduct a breadth-first search on the scaled graph. The algorithm finds a route that has a score being no worse than $\frac{1}{1-\epsilon}$ of the optimal route.

4.4 Database Management Systems

Database management systems [5, 7–9] such as MySQL, PostgreSQL, MongoDB, and Oracle are able to support spatial or keyword search. They use spatial and textual indices separately to organize the geo-textual data. In terms of spatial indexing, most of the systems support the R-tree. In terms of textual indexing, they all support the inverted index. None of them support hybrid geo-textual indices. Thus, existing database management systems support spatial keyword queries by using spatial and textual indices separately. However, they are not as efficient as algorithms that use hybrid geo-textual indices as they do not fully utilize the filtering on both the spatial and textual attributes.

4.5 Other Software Libraries

Apache Lucene [2] is an open-source, full-text search engine. It provides APIs to support spatial search queries on document collections, which are similar to the standard spatial keyword queries, such as BRSK, BkSK, and TkSK. It employs separate indices for spatial attributes, including points and other shapes, and textual features. In particular, Lucene implements a KD-tree variant called the block KD-tree, which is designed to enable efficient IO, and Lucene also implements a multi-level grid structure for Geohash. Next, Lucene supports inverted files for textual attributes. Location based services such as Google Maps and Foursquare also support spatial queries on POIs that are similar to the standard spatial keyword queries. It appears that they index spatial and textual features separately. Both services base their spatial indexing on the S2 Geometry Library [3], which assigns data to so-called S2 cells and enable search on such cells. S2 cells are obtained by mapping every point on the Earth to one of the six faces of an enclosing cube. Each face is than partitioned Quad-tree style into a hierarchy of cells. The cells at all levels are enumerated using a Hilbert Curve, so that every S2 cell has a unique identifier.

5 Problem Definition for Querying Streaming Geo-textual Data

We organize existing studies according to four categories: publish/subscribe systems, localized event detection, temporal spatial keyword queries, and location-based term queries. Table 8 gives a brief introduction to each type of work.

5.1 Publish/Subscribe Systems

The techniques for managing static geo-textual data employ a user-initiated model (a.k.a. a "pull" model), where a user issues a query and the system responds with desired answers. However, such a model is not always suitable for querying steaming data because it does not provide users with real-time answers. This motivates a server-initiated model (a.k.a., a "push" model) where users register subscriptions, also called continuous or standing queries, that are evaluated continuously against the streaming data so that results can be pushed back to the users in real time. Consequently, a high-performance scalable publish/subscribe system is required that is able to evaluate continuously a set of subscriptions against incoming data.

Query Type	**Description**
Publish/subscribe systems [10, 30, 31, 42, 72, 85, 107, 108, 138, 140, 155, 156]	Publish/subscribe systems allow users to submit subscriptions that specify spatial and textual matching conditions. Such systems will notify the users in real-time when incoming data satisfies the matching conditions.
Localized event detection [11, 35, 57, 82, 89, 118, 124, 126, 137, 141, 157, 160, 165]	A local event is typically a bursty activity that occurs in a local area in a specific timespan. These studies investigate how to represent local events and extract local events efficiently from streaming geo-textual data.
Temporal spatial keyword queries [14]	This type of query finds objects that satisfy spatial, textual, and temporal constraints. The queries are similar to previously covered Euclidean space queries except that they target streaming data.
Location-based term queries [105, 131, 139, 151]	This type of query focuses on term frequencies in object streams. Objects are filtered by spatial and textual constraints, upon which term frequencies are extracted from the remaining objects.

Table 8 Summary of Studies on the Querying of Streaming Data

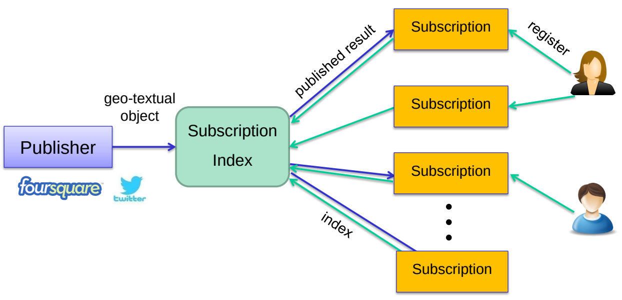


Fig. 3 Framework for a Spatial Keyword Publish/Subscribe System.

Figure 3 illustrates the general framework of a spatial keyword publish/subscribe system over a geo-textual data stream. It can be modeled as a system that manages a stream of incoming geo-textual objects (e.g., geo-tagged Tweets) generated by a publisher (e.g., Twitter) and a large number of subscriptions. Each spatial keyword subscription contains a spatial argument, a textual argument, and possibly a temporal argument, which lead to three kinds of matching conditions: (1) spatial, (2) textual, and (3) temporal matching conditions. When a new object arrives, the system sends the object to the subscriptions whose matching conditions are satisfied by the object.

Publish/subscribe systems for textual data have been studied widely (e.g., [67, 104, 129]) without taking into account the location aspect. The problem of processing spatial keyword subscriptions has been considered in a number of studies.

We proceed to classify the subscriptions based on their matching conditions. Table 9 presents a summary of the categorization.

Spatial matching conditions. We classify spatial matching conditions into range matching and distance matching. Range matching is the dominant spatial matching condition in existing studies [10, 30, 42, 63, 85, 107, 108, 136, 140, 155, 156]. The spatial component in these studies is a region. Consider a new geo-textual object d in a geo-textual stream. If d has a point location, d satisfies the spatial matching condition if its location belongs to the query region; If d has a region location, d satisfies the spatial matching condition if its region overlaps with the query region [85, 155, 156].

Next, studies also exist that support distance matching. When the spatial component in a subscription is a point location, some studies [31, 37, 72, 73, 138] use the spatial proximity between the subscription and the location of a geo-textual object for ranking. When the spatial component is a region, one study [156] uses the spatial overlap between the subscription and the region of a geo-textual object as a matching score.

The range matching corresponds to the range query in spatial databases, while the distance matching corresponds to the nearest neighbor query in spatial databases.

Textual matching conditions. Textual matching is either treated as a Boolean condition or is used for ranking. In Boolean matching, the new document d of an object either matches or does not match the textual component of a subscription q. Specifically, Li et al. [85], Guo et al. [63], Wang et al. [140], Yu et al. [155], Chen et al. [42], and Mahmood et al. [107] use Boolean AND semantic, i.e., for d to match q textually, d must contain all the query keywords in q. The frameworks developed by Chen et al. [30], Mahmood et al. [108] and Abdelhamid et al. [10] support both Boolean AND and Boolean OR semantics. Next, another line of work [31, 37, 72, 136, 156] computes a text similarity score for the match between q and d. The score is combined with a spatial matching score to determine whether d is a result of q.

Boolean matching corresponds to a Boolean query in information retrieval, while the ranking matching corresponds to a ranking query in information retrieval.

Temporal matching conditions. In addition to the spatial and textual aspects, several proposals consider

Literature	**Approach Abbr.**	**Spatial Match**	**Textual Match**	**Temporal Match**	**Query Type**
Abdelhamid et al. [10]	Cruncher	range, distance	Boolean (`AND, OR`)	none	RB, kB
Chen et al. [30]	IQ-tree	range	Boolean (`AND, OR`)	none	RB
Chen et al. [31]	TaSK	distance	similarity	decay function	$\text{DS}_{rank,decay}$
Chen et al. [42]	PS^{2}Stream	range	Boolean (`AND`)	none	RB
Hu et al. [72]	Stamp	distance	similarity	none	$\text{DS}_{threshold}$
Li et al. [85]	R^t-tree	range	Boolean (`AND`)	none	RB
Mahmood et al. [107]	FAST	range	Boolean (`AND`)	none	RB
Mahmood et al. [108]	Tornado	range, distance	Boolean (`AND, OR`)	none	RB, kB
Wang et al. [138]	SKYPE	distance	similarity	sliding window	$\text{DS}_{rank,sliding}$
Wang et al. [140]	AP-tree	range	Boolean (`AND`)	none	RB
Yu et al. [155]	MBRTrie, PT-Quadtree	range	Boolean (`AND`)	none	RB
Yu et al. [156]	R^t-tree	range, distance	Boolean (`AND`)	none	RB,$\text{DS}_{threshold}$

Table 9 Comparison of Existing Work on Spatial Keyword Subscription Query

temporal aspects in their matching conditions. These proposals [31, 37, 136, 138] use either decaying schemes or sliding windows in their temporal matching conditions. Decaying schemes make it possible to take into account the freshness of objects in rankings that aim to maintain the k most relevant objects for a subscription. Specifically, Chen et al. [31] apply an exponential decaying function to quantify the freshness of a geo-textual object, which is incorporated into the ranking of geo-textual objects. Next, Wang et al. [138] and Wang et al. [136] use sliding windows in their temporal matching. In particular, they continuously maintain top-k geo-textual objects within a sliding window based on the spatial and textual relevance between an object and a subscription.

Combined matching conditions. In existing proposals, combined spatial and textual matching scores are usually used to determine whether a geo-textual object matches a query. Combinations of three types of matching conditions are used in the literature. First, some studies [10, 30, 42, 85, 107, 108, 140, 155, 156] combine range matching with Boolean matching: A geo-textual object d matches a subscription q if d's location belongs to the region of q or if d's region overlaps with the region of q, if d's spatial information is a region AND if d's textual component satisfies a Boolean condition w.r.t. q's textual component. Next, two demonstration systems, Cruncher [10] and Tornado [108], maintain the k nearest geo-textual objects to each of a set of subscriptions among those objects whose textual component satisfies the subscription's Boolean matching condition. In yet another line of work [31, 72, 138, 156], spatial distance matching and textual similarity are combined. The combination takes two different forms: (a) Ranking. The subscription query q considered by Chen et al. [31], Wang et al. [138], and Yu et al. [156] continuously maintains k most relevant geo-textual objects based on a scoring function that combines spatial proximity and textual similarity. In particular, if a new object d is one of the k most relevant objects to a query q, d is a match for q. (b) Threshold. Hu et al. [72] consider a different type of subscription that finds results based on a pre-specified threshold θ on the score. Specifically, if the score between d and q exceeds θ, we say d matches q.

Based on different combinations of spatial and textual matching as well as the incorporation of temporal matching, we name the existing subscription queries as follows: (1) RB query. This query combines spatial range matching and Boolean matching for textual component. (2) kB query. This query combines spatial kNN matching condition and Boolean matching for textual component. (3) DS query. This query combines spatial distance matching and textual similarity scoring. We denote the two types of combination, i.e., ranking and threshold, by DS_{rank} and $\text{DS}_{threshold}$, respectively. In the literature, DS_{rank} queries are further combined with the two types of temporal matching, yielding two types of DS_{rank} queries, denoted by $\text{DS}_{rank,decay}$ and $\text{DS}_{rank,sliding}$. We note that not all combinations of the spatial and textual matching conditions are considered in previous work. It might be of interests to consider the practicality and feasibility of unexplored combinations in future work.

Other subscription queries. Guo et al. [63] consider the problem of continuously monitoring moving users subscribing to streaming geo-textual data. This query can be considered as an extension of the RB subscription query that allows queries to move. Chen et al. [37] extend the $\text{DS}_{rank,decay}$ query by returning k clusters of geo-textual objects, rather than k geo-textual objects. Wang et al. [136] study subscription queries that return top-k objects over a sliding window, considering also the credibility and representativeness of returned objects.

5.2 Localized Event Detection

As suggested already, a localized event is typically a bursty activity that occurs in a local area during a specific timespan, such as a demonstration, a conference, a sports match, or an emergency response. A host of studies exist on the problem of detecting local events from geo-tagged data streams. We classify these studies based on the five aspects covered next. A summary of the categorization can be found in Table 10.

Local event representation. Existing proposals represent a local event differently, and we classify them into four categories.

(1) *Cluster of geo-tagged micro-blog posts*: A local event is represented as a cluster of geo-tagged micro-blog posts. Zhang et al. [157, 160, 165] represent an event by a geo-topic cluster, which is a set of geo-tagged micro-blog posts whose locations are close to each other and whose text has similar meanings. The meanings of a cluster should deviate from those of routine activities. Watanabe et al. [141] represent an event as a cluster of micro-blog posts that are geographically and temporally close to each other. Sankaranarayanan et al. [126] find local events by clustering geo-textual objects based on their spatial proximity and textual relevance. Each local event is represented by a cluster of micro-blog posts that are spatially and textually similar to each other. Li et al. [89] represent a criminal event as a cluster of textually similar micro-blog posts within a specified region. Sakaki et al. [124] detect earthquakes using groups of micro-blog posts that are posted during a period—They build classifiers to determine whether posts are earthquake related and whether a group of posts corresponds to an earthquake.

(2) *Vector*: Quezada et al. [118] represent an event based on the geographical distribution (over countries) of geo-textual objects that cover a given topic. Specifically, an event is defined by a *protagonism-participation vector*, where *protagonism* captures countries where events originated and *participation* captures countries where people talk about the events.

Feng et al. [57] represent an event as a group of hashtags. A hashtag h is a vector that is composed of a normalized word weight vector and a normalized tag weight vector:

$$h = (\alpha^{0.5} h_{word}, \beta^{0.5} h_{tag}),$$

$$h_{word} = (w_1, w_2, ..., w_{|W|}),$$

$$h_{tag} = (h_1, h_2, ..., h_{|H|}),$$

where w_i is the weight of the i-th word, $|W|$ is the number of words, h_i is the weight of the i-th tag, and $|H|$ is the number of tags.

(3) *Summary*: Krumm et al. [82] find local events by detecting anomalies in terms of the number of tweets at different spatial and temporal resolutions. An anomaly gives rise to a space-time prism (S, T), which indicates the location and time of a local event. Five tweets in a prism (S, T) are extracted as a summary representation of the corresponding event.

(4) *Terms/Tags*: Yang et al. [137] represent events as local maximal frequent keyword patterns (LMFP). To define such patterns, we define a local frequent keyword pattern (LFP). Given a data stream D, a region R, and a threshold θ, a pattern P is an LFP if its local frequency $f(P, R) \geq \theta$. The local frequency $f(P, R)$ is computed by

$$f(P, R) = |\{o | o \in D_R \wedge P \subseteq o.\psi\}|,$$

where D_R represents the set of objects in R and $o.\psi$ represents the textual information of o. An LFP P is an LMFP if P is not the subset of any other LFP, i.e., $\nexists P' \supset P$ where P' is an LFP.

Abdelhaq et al. [11] use a cluster of terms occurring in geo-textual posts that are close in spatial distance to represent an event. Chen et al. [35] use a cluster of tags to represent an event, where the tags are clustered based on their co-occurrence and spatial and temporal distributions.

Event type. We divide existing proposals into two categories: (1) detecting a particular type of local event (e.g., an emergency or an earthquake) and (2) extracting various types of event.

Real-time. We divide existing proposals into two categories: (1) detecting and monitoring local events in a real-time fashion, denoted by "Online - yes," and (2) extracting local events in batch mode.

Number of events. We divide existing studies into two categories: (1) studies that perform top-k selection to identify local events from a set of event candidates, denoted by "Pre-specified," and (2) studies that do not need to specify "k" as input, which is denoted by "Undefined."

Temporal constraint. Zhang et al. [157,160,165] and Abdelhaq et al. [11] detect local events from streaming geo-textual objects in a sliding window. Li et al. [89] extract local crime events in a specified timespan. Watanabe et al. [141], Quezada et al. [118], Sakaki et al. [124], Chen et al. [35], and Krumm et al. [82] do not consider a time constraint.

Literature	Approach Abbr.	Local Event Representation	Event Type	Online	# Events	Temporal Const.
Abdelhaq et al. [11]	EvenTweet	geo-cluster of terms	general	yes	pre-specified	sliding window
Chen et al. [35]	n.a.	geo-temporal cluster of Flickr tags	general	no	undefined	none
Feng et al. [57]	StreamCube	vector of terms and tags	general	yes	undefined	timespan
Krumm et al. [82]	Eyewitness	summary	general	no	undefined	none
Li et al. [89]	Tedas	textual cluster within a region	crime	no	undefined	timespan
Quezada et al. [118]	n.a.	protagonism-participation vector	general	no	pre-specified	none
Sakaki et al. [124]	n.a.	tweet cluster	earthquake	yes	undefined	none
Sankaranarayanan et al. [126]	TwitterStand	geo-textual cluster	general	yes	undefined	none
Watanabe et al. [141]	Jasmine	geo-temporal cluster	general	no	undefined	none
Yang et al. [137]	n.a.	frequent keyword pattern	general	yes	undefined	sliding window
Zhang et al. [157]	TrioVecEvent	geo-topic cluster	general	yes	undefined	sliding window
Zhang et al. [160]	GeoBurst	geo-topic cluster	general	yes	pre-specified	sliding window
Zhang et al. [160]	GeoBurst+	geo-topic cluster	general	yes	undefined	sliding window
Zhang et al. [165]	Event-Radar	geo-topic cluster	general	yes	undefined	sliding window

Table 10 Comparison of Existing Studies on Local Event Detection

5.3 Temporal Spatial Keyword Query

Almaslukh et al. [14] investigate the problem of processing temporal spatial keyword queries over streaming geo-textual data. Such queries define constraints on the temporal, spatial, and textual attributes of the data. Let D denote a set of streaming geo-textual objects. Each object $o \in D$ is a triple (ρ, ψ, t), where $o.\rho$ is a spatial point, $o.\psi$ is a set of keywords, and $o.t$ is a timestamp. Let D_t be the snapshot of dataset D at time t, such that every object $o \in D_t$ has $o.t \leq t$. The study considers the processing of two types of queries.

Temporal Boolean range spatial keyword (TBRSK) query. This query uses a set of keywords and a spatial region as filters, and it retrieves the most recent objects with respect to the query time. Formally, a TBRSK query $q = (t, w, r, k)$ has four arguments: $q.t$ is a timestamp, $q.w$ is a set of keywords, $q.r$ is a spatial region, and $q.k$ is an integer. The result $q(D)$ of q consists of k objects from D satisfying that $q(D) \subseteq q(D_t)$, where $\forall o \in q(D_t)\,(o.\rho \in q.r \wedge o.\psi \cap q.w \neq \emptyset \wedge o.t \leq q.t)$, and $\forall o \in q(D)\,(\forall \hat{o} \in q(D_t) \backslash q(D)\,(o.t > \hat{o}.t))$.

Temporal Boolean top-k spatial keyword (TBkSK) query. This query uses a set of keywords as a textual filter and finds top-k objects according to a score function considering spatial proximity and temporal difference. Formally, a TBkSK query $q = (t, w, \rho, k)$ has four arguments: $q.t$ is a timestamp, $q.w$ is a set of keywords, $q.\rho$ is a spatial point, and $q.k$ is an integer. The result $q(D)$ of q consists of k objects in D that satisfies $q(D) \subseteq q(D_t)$, where $\forall o \in q(D_t)\,(o.\psi \cap q.w \neq \emptyset \wedge o.t \leq q.t)$, and $\forall o \in q(D)\,(\forall \hat{o} \in q(D_t) \backslash q(D)\,(F(o, q) \leq F(\hat{o}, q))$. Here, function $F(o, q)$ computes the spatial-temporal relevance of an object o to q:

$$F(o, q) = \alpha \cdot \frac{dist(o.\psi, q.\rho)}{R_{max}} + (1 - \alpha) \cdot \frac{q.t - o.t}{T_{max}},$$

where α is a weight parameter, R_{max} is the maximum spatial distance, and T_{max} is the maximum temporal difference.

5.4 Location-based Term Queries

Location-based term queries focus on the frequencies of terms in streaming geo-textual objects. Existing studies belong to four categories, as explained next.

Top-k most frequent terms query [131]. This query finds top-k most frequent terms over streaming geo-textual objects given a region and a timespan. Let D be a stream of timestamped geo-textual objects. The frequency of a term w for a set of objects O is the number of objects in O whose textual information contains w. A top-k most frequent terms query is given by $q = (R, T, k)$ where R denotes a spatial region, T denotes a time interval $[t_s, t_e]$, and k is the result cardinality. The query returns the k most frequent terms in the objects $O_q \subseteq D$ that fall in R and whose timestamps belong to T.

Top-k most trending terms query [105]. Given trending measure, this query finds the k most trending terms with a region and a time interval. GeoTrend [105] defines a query by a quadruple $q = (R, T, k, \mathit{Trend})$, where R is a spatial region, T is a number of time units, k is the result cardinality, and *Trend* is a trending measure. At each time unit, the frequency of a term w is the number of objects whose textual information contains w. GeoTrend finds k terms such that: (1) the terms occur in objects that are in R, (2) the terms occur in objects whose timestamp belongs to the T time units, and (3) the terms are the top-k ranked terms based on the *Trend* measure among all terms in objects in R and T.

Location-based top-k term (LkT) query [151]. Given an LkT query $q = (\rho, k)$, where ρ is a query location and k is the result cardinality, the query finds k terms with the highest location-aware frequency among the terms occurring in objects in a sliding window W. The location-aware frequency score of a term t is a linear combination of the distances between the geo-textual objects containing t and the query location and t's frequency, as formalized in Equation 14:

$$\begin{aligned} ST(t,q) =& \alpha \cdot \frac{|W_t|}{|W|} + \\ & (1-\alpha) \cdot (1 - \frac{\sum_{o \in W_t} dist(q,o)}{d_{diag} \cdot |W_t|}), \end{aligned} \tag{14}$$

where $\alpha \, (0 \leq \alpha \leq 1)$ is a user parameter, W_t denotes the set of objects in W containing term t, and d_{diag} denotes the diagonal length of the minimum bounding rectangle (i.e., MBR) of W_t.

Selectivity estimation [139]. Wang et al. [139] investigate the problem of estimating the cardinality of geo-textual objects in a stream whose location falls in a specified region and textual information satisfies a specified Boolean expression. Conjunction (AND), disjunction (OR), and negation (NOT) semantics are taken into consideration.

6 Querying Streaming Geo-Textual Data - Methodology

6.1 Publish/Subscribe Systems

The main challenge to enable efficient processing of subscription queries over spatio-textual data streams is how to organize a large number of subscription queries to facilitate the efficient processing of incoming spatio-textual objects. Specifically, existing location-aware publish/subscribe proposals focus on the *document-queries matching* problem, which aims to find efficiently the subscription queries that match incoming geo-textual objects.

Existing solutions can be classified into centralized [30, 31, 37, 63, 72, 73, 85, 107, 136, 138, 140, 155, 156] and distributed solutions [10, 40, 42, 106, 108]. The former focus on inventing effective index structures for subscription queries. The distributed solutions utilize a cluster of servers and consider workload partitioning that enables the system to achieve high performance, e.g., large throughput. They also consider workload adjustment strategies that adapt to changing data distribution.

6.1.1 Centralized Solutions

Existing solutions propose different indices to organize the subscription queries based on their spatial and textual attributes. Table 11 classifies the existing subscription query indexing schemes.

Indexing priority. Most proposals [30, 31, 37, 63, 72, 73, 85, 107, 136, 138, 156] use *spatially-prioritized* indexing schemes: subscriptions are first partitioned by their spatial attribute and then by their textual attribute. Specifically, they organize the subscriptions using a spatial tree structure (e.g., R-tree, Quad-tree) and embed textual indexing (e.g., inverted index) in the nodes of the spatial index . Such spatially-prioritized indices are more efficient for subscriptions with high spatial selectivity, i.e., subscriptions with small spatial regions.

Textually-prioritized indices first organize subscription queries by their textual attribute and then by their spatial attribute. The only such indexing scheme is the MBRTrie [155], a trie where each trie node is associated with a query keyword. Textually-prioritized indices are more efficient for subscriptions with high textual selectivity, e.g., subscription queries with few and highly selective keywords.

Going beyond the spatially- and textually-prioritized indices, *hybrid* indices (e.g., AP-tree [140]) are capable of adaptively prioritizing the spatial or textual aspects by means of a cost model.

Spatial indexing scheme. For the purpose of spatial indexing, existing methods employ variants of popular spatial indices. Most proposals use Quad-tree-based partitioning (e.g., [30, 31, 37, 63, 107, 138, 140]) or R-tree-based partitioning (e.g., [72, 73, 85, 156]). Wang et al. [136] use Grid-index-based partitioning. Overall, the R-tree is the most powerful at pruning, while the Quad-tree and Grid index are more efficient for update (i.e., inserting or deleting subscriptions). Therefore, Quad-tree and Grid-index-based partition schemes are more suitable for applications with frequent subscription updates.

Textual indexing scheme. MBRTrie [155] is a textually-prioritized index that uses a trie to organize the subscription queries. Each trie node is associated with a query keyword, and the path from the root to any node represents a unique keyword sequence. The index imposes a global keyword order, so when inserting a query, its keywords are considered in order to determine the next-level node; a new node is created if no

Literature	**Query Index Name**	**Indexing Priority**	**Spatial Indexing Scheme**	**Textual Indexing Scheme**
Chen et al. [30]	IQ-tree	Spatially-prioritized	Quad-tree	Inverted file
Chen et al. [31]	CIQ-tree	Spatially-prioritized	Quad-tree	Inverted file
Chen et al. [37]	Quad-tree + Inverted file	Spatially-prioritized	Quad-tree	Inverted file
Guo et al. [63]	BEQ-tree	Spatially-prioritized	Quad-tree	Inverted file
Hu et al. [72]	Stamp	Spatially-prioritized	R-tree	Inverted file, Summary file
Hu et al. [73]	R^I-tree	Spatially-prioritized	R-tree	Interval tree
Li et al. [85]	R^t-tree	Spatially-prioritized	R-tree	Inverted file, Summary file
Mahmood et al. [107]	FAST	Spatially-prioritized	Quad-tree	EKI
Wang et al. [136]	GH	Spatially-prioritized	Grid	Inverted file, Summary file
Wang et al. [138]	Quad-tree + Inverted file	Spatially-prioritized	Quad-tree	Inverted file
Wang et al. [140]	AP-tree	Hybrid	Quad-tree	f-ary tree
Yu et al. [155]	MBRTrie	Textually-prioritized	MBR	Trie & Inverted file
	PT-Quadtree	Spatially-prioritized	Quad-tree	Summary file
Yu et al. [156]	R^t-tree	Spatially-prioritized	R-tree	Inverted file, Summary file

Table 11 Subscription Query Indexing Schemes

matching node exists. Each query is stored in a node such that the path from the root to that node matches the query's keyword sequence.

Most other proposals employ a spatially-prioritized index that maintains an inverted index in the nodes of the spatial index (or cells if a Grid index is used). Specifically, the IQ-tree [30] and CIQ-tree [31], which are proposed for indexing Boolean-based spatial keyword subscriptions and similarity-based spatial keyword subscriptions, respectively, are basically Quad-trees where each node is augmented with an inverted index on the queries assigned to the node. A cost-model-based algorithm is developed for finding the node(s) in which to store a query by considering the trade-off between the costs of index updates and queries.

Mahmood et al. [107] propose a textual index, called the expandable keyword index (EKI), which is integrated in the nodes of a Quad-tree-based index. EKI is based on the trie and considers the frequencies of query keywords. Each node is labeled by a query keyword. Queries are first stored in the top-level nodes according to their least frequent keyword. When the size of a node reaches a threshold, it is expanded by creating a child node that is labeled by another query keyword.

Other spatially-prioritized indices [72, 85, 156] use a summary file for the textual indexing. Such files contain a set of tokens that are selected from the query keywords, and possibly corresponding weights.

Indexing subscription queries. We use the IQ-tree as a representative in order to explain the procedure of indexing subscription queries. The IQ-tree targets Boolean-based spatial keyword subscriptions. It extends a Quad-tree by augmenting each node with an inverted index. When inserting a subscription, it uses a cost model to identify an appropriate node or nodes among the nodes that cover or overlap the subscription's range. The cost model considers the keyword distribution in the spatial range of the nodes, the goal being to achieve optimal query and update performance. When a geo-texutal document arrives, the search for matching subscriptions starts at the root node, and the search procedure recursively checks the inverted indices of the nodes that cover the document location. For each keyword in the document, the subscriptions in the corresponding posting list are checked to see whether the new document matches them.

6.1.2 Distributed Solutions

There are two main components in distributed frameworks for querying geo-textual data streams: router and worker. A router has a global index that partitions the incoming workload among workers. A worker has a local index to facilitate the matching between geo-textual documents and subscriptions, which is similar to the centralized solutions. The distributed solutions support workloads composed of processing geo-textual documents and inserting and deleting subscriptions. Unlike the centralized solutions, they focus on workload partitioning. Multiple factors such as load balance and network cost need to be considered. The design of the global index is core to the workload partitioning. Therefore, we classify existing distributed solutions based on the global index they employ.

KD-tree. A spatial-based partitioning strategy partitions a workload based on the spatial attribute. Tornado [108] and Cruncher [10] adopt a KD-tree in the router. The KD-tree recursively partitions the space by alternating between the x-dimension and y-dimension at each tree level, and each leaf node is assigned to a worker. To assign a geo-textual document d to a worker, the router finds the leaf node that d falls into and then sends d to the corresponding worker. To distribute a subscription update request q (insert or delete), the router finds the leaf nodes that q intersects and sends q to the corresponding workers.

Augmented-Grid. A new version of Tornado [106] adopts a structure called the Augmented-Grid (A-Grid, for short) to partition a workload. A-Grid first splits the space into virtual fine-grained grid cells *FG*. It then partitions the space into partitions that are overlaid on top of *FG*. The router assigns each such partition to a worker and maintains a summary of the query keywords for each worker.

Grid^t. Chen et al. [42] propose a hybrid workload partitioning strategy that leverages both the spatial and the textual attribute to partition a workload. The motivation is that the data distributions in different regions are different and that adopting textual partitioning in some regions can enhance the filtering of the router. An index called the grid^t-tree is proposed, which is a Grid index where cells are further partitioned by the textual attribute.

QT-tree. Chen et al. [40] propose a hybrid index called the QT-tree. It is a variant of the Quad-tree that allows a node to be split based on the spatial or textual attribute of the data. It aims to minimize the total workload while balancing the load among the workers. To achieve this, a cost model is used to decide between using spatial or textual partitioning when building the QT-tree.

6.2 Localized Event Detection

Existing localized event detection proposals can be classified into detecting general events and detecting domain-specific events. The main difference is that for general event detection, no knowledge of the kinds of events to be detected is assumed in advance, while for domain-specific event detection, the type of event to be detected is known. Events are usually represented as clusters of geo-textual objects. When detecting general events, a popular scheme [57, 118, 126, 157, 160, 165] is to first generate candidate events by clustering the objects based on their spatial and textual attributes, and then use a classifier to eliminate non-event clusters. Another approach [11,35,82,137,141] is to first find regions where abnormal patterns occur, e.g., an unexpected spike in some words, and then cluster the objects in those regions to obtain events. Considering the methods for detecting domain-specific events [89, 124], the main idea is to train and use a classifier to judge whether incoming objects are relevant to a specific event. An event is then characterized by its set of relevant objects.

6.2.1 General Event Detection

Some proposals find candidate events by conducting online clustering and then apply filtering to remove non-event clusters. We call this as clustering-and-filtering. Another approach is to find regions where abnormal patterns happen and then cluster the geo-textual objects in those regions to obtain events. We call this as checking-and-clustering, where the *checking* implies that we need to check the objects in a region to decide whether an abnormal pattern occurs. We consider the two approaches in turn.

Clustering-and-filtering. Studies in this category differ in how they perform clustering: TrioVecEvent [157] uses a trained model to learn multi-modal embeddings of geo-textual objects and then performs online clustering using a Bayesian mixture model. GeoBurst [160] and Event-Radar [165] conduct online clustering by determining a set of pivot geo-textual objects and assigning new objects to pivot objects to produce clusters. StreamCube [57] uses a hierarchical index to categorize the geo-textual objects, forming clusters of objects that are temporally and spatially close and that have the same hashtag. Finally, TwitterStand [126] represents geo-textual objects as vectors and place objects with high cosine similarity in the same cluster, which is split into smaller clusters based on the spatial attribute.

Checking-and-clustering. Studies in this category differ in how to find an abnormal pattern that is used for recognizing an event. Chen and Roy [35] detect events from geo-tagged photos. They analyze the temporal and location distribution of the tags and find the tags that show significant distribution patterns (e.g., burstiness), which are then clustered to represent events. EvenTweet [11] finds abnormal patterns by finding words having a bursty frequency. For each bursty word, the corresponding region is found, and a set of representative words in the region is used to represent the event. Watanabe et al. [141] collect groups of geo-textual objects that are spatially and temporally close. For each such group, co-occurring words are used to represent

Index Name	First Level	Second Level
Grid-inverted	Grid index	Inverted index
Inverted-grid	Inverted index	Grid index
Inverted-quadtree	Inverted index	Quad-tree
Inverted-Rtree	Inverted index	R-tree
Quadtree-inverted	Quad-tree	Inverted index
Rtree-inverted	R-tree	Inverted index

Table 12 Six Types of Hybrid Indices

the corresponding event. Krumm and Horvitz [82] discretize space and time into space-time pieces and find abnormal patterns by identifying pieces with an anomalous spike. For each piece found, they extract tweets to represent the corresponding event. Yang et al. [137] find abnormal patterns by identifying local maximal frequent keyword co-occurrence patterns, which are used to represent events in different regions.

6.2.2 Domain-specific Event Detection

To detect domain-specific events, e.g., crimes or an earthquake, existing methods train a classifier and employ it to check whether new geo-textual objects are related to the event. They also provide methods to explore the properties of events, e.g., their spatial distribution. Existing proposals differ mainly in the classifier they used.

Sakaki et al. [124] target earthquake detection. They devise and train a classifier using a support vector machine. The classifier is used to judge whether incoming geo-textual tweets are relevant to an earthquake. They design a temporal model to decide when to issue an earthquake alert after having collected a certain amount of earthquake relevant tweets. They also propose a spatial model to estimate the location of an earthquake. Tedas [89] is developed for detecting Crime and Disaster relevant Events (CDE) from Twitter. A classifier is employed that is trained based on Twitter- and CDE-specific features.

6.3 Temporal Spatial Keyword Queries

Almaslukh and Magdy [14] evaluate the performance of 10 different indices that can be employed to process the temporal spatial keyword queries. These include three spatial indices, one textual index, and six hybrid indices. The spatial indices are the Grid index, the Quad-tree, and the R-tree. The only textual index is the inverted index. The hybrid indices employ two levels of indexing, spatial-based indexing and text-based indexing. Table 12 presents an overview.

Processing TBRSK queries. Processing the TBRSK queries involves three steps. The first step is to use the index to access the indexing entries (e.g., the leaf nodes of the Quad-tree) based on a query range or query keywords. The entries are placed in a queue. The second step is to traverse the entries and access the objects stored in them. An initial list L of k objects is created. The third step is to traverse the remaining entries in the queue and compute the final result. During this step, L is kept up to date and is used for temporal pruning: if the largest timestamp of the objects in an entry is smaller than the smallest timestamp in L, the entry can be pruned. The pruning can be done efficiently as the objects are sorted in ascending order of their timestamp.

Processing TBkSK queries. Again, the query processing involves three steps. The first step is to use the index to retrieve the indexing entries based on the query location and keywords. An initial list L of k objects is again created. In the second step, an upper bound on the distance to the query location is computed based on the scoring function and L. The entries within the distance upper bound are then traversed and put in a queue. The third step is to traverse the entries in the queue to compute the final result. Here, spatial and temporal pruning strategies can be used for pruning.

6.4 Location-based Term Queries

Top-k most frequent terms query. Skovsgaard et al. [131] propose an Adaptive Frequent Item Aggregator (AFIA) system to answer the top-k most frequent terms query. AFIA employs a multi-granularity grid index. For each granularity, it uses a grid index with a fixed cell size. Each cell maintains counters of the most frequent terms. The number of counters in each cell can be varied with the time elapsed, which is achieved by aggressive increment and relaxed decrement operations. When a cell receives many inserts, it will perform an aggressive increment to increase the number of counters. On the other hand, a relaxed decrement is performed if a cell sees little activity for a while. A query is processed by merging the counters in the cells that the query region overlaps with.

Top-k most trending terms query. GeoTrend [105] employs a multi-granularity grid index to find the top-k most trending terms in a dynamic data set. Each index cell maintains a trending terms list, which is achieved by storing the frequency of terms in the most recent T time units, where T is a system parameter. To process a trending query with a spatial region R, GeoTrend starts at the root cell, which has the coarsest granularity, and recursively accesses children (i.e., finer-granularity cells) that overlap with R. A cell is a candidate for further processing if it is a leaf cell (i.e., a cell having the finest granularity) or is completely covered by R. GeoTrend merges the top-k most trending terms maintained in all candidate cells to compute the result. The assumption is that the final top-k most trending terms must have appeared in at least one top-k list in those cells. When the assumption does not hold, the result is inaccurate.

Location-based top-k term (LkT) query. Xu et al. [151] use a Quad-tree-based index to solve the LkT query. Each node in the Quad-tree maintains a summary of the term information of objects in the node. When an object arrives, the leaf node containing the object is found, and then the textual summary in that node is updated. Then upward summary merging operations are performed that update the textual summary of the parent nodes by merging the summaries stored in the child nodes. To answer an LkT query, the Quad-tree is traversed, and the scores of the terms are computed based on the textual summaries. Intermediate results are put into a priority queue, and the procedure terminates when top-k results are found.

Selectivity estimation. Wang et al. [139] study the problem of estimating the selectivity of a spatial keyword query given by $q = (R, T)$, where R is a region and T is a set of keywords. The result consists of all objects that fall into R and that contain all keywords in T. Two baseline algorithms are proposed: The Adaptive Space Partitioning (ASP) Tree [70] based algorithm and the k Minimal Values (KMV) Synopses [17] based algorithm. Evidence is provided to the effect that the ASP-tree-based algorithm is preferable when T contains at most one keyword and that the KMV-based algorithm is otherwise better. Wang et al. also propose the A^{2}SP structure that combines the ASP tree with KMV synopses. If T contains only one keyword or keywords that do not have an ASP-tree due to low frequency, the ASP-tree-based algorithm is used. Otherwise, a local Bayesian network is used to learn the local correlations among keywords in T within region R, and to derive the selectivity estimate on the KMV synopses.

7 Conclusions and Future Work

This paper provides a comprehensive survey of problem formulations and solutions of studies on the querying of geo-textual data. It classifies studies into ones querying static geo-textual data and ones querying streaming geo-textual data, and it reviews the problem formulations and solutions for these classes.

Querying static geo-textual data. This category of studies can be categorized into querying in Euclidean space versus in road networks, the main difference being the way of computing distances between spatial objects. Although the problem definitions are similar across the two categories, the proposed solutions have notable differences.

Not surprisingly, standard queries are the most common and are well studied: the BRSK, the BkSK, and the TkSK queries for querying in Euclidean space, and the BRSK$_{\text{road}}$ and TkSK$_{\text{road}}$ queries for querying in road networks. To handle cases where users have requirements that go beyond the spatial and textual aspects, studies target different extensions to the standard queries that incorporate support for, e.g., temporal or social aspects, examples including the TBRSK, Social_TkSK, and MBkSK$_{\text{road}}$ queries. Going beyond single-object granularity queries, studies also consider group queries that take into account inter-relations among objects to retrieve groups of objects that combine to form query answers. Studies also exist that target query modifications to serve users who are unable to provide "good" spatial keyword query parameters up front. These studies enable the refinement of query parameters so that users are satisfied with the query results. Additionally, studies exist on spatio-textual joins and region finding/analysis queries. A prominent application in road networks is route planning, where the aim is to find an optimized route in a road network that satisfies given spatial and textual requirements.

Querying streaming geo-textual data. We classify the studies on querying streaming geo-textual data into four categories: publish/subscribe systems, localized event detection, temporal spatial keyword queries, and location-based term queries. Publish/subscribe systems allow users to register subscriptions that specify spatial and textual matching conditions and possibly temporal matching conditions. Users are then notified when incoming data objects match the conditions stated in their subscriptions. Studies on local event detection consider the problem of detecting local events from streaming geo-textual data. A local event can be represented by a cluster of objects, a vector, a summary, or a set of terms or tags. The studies on temporal spatial keyword queries focus on TBRSK and TBkSK queries, both of which are modifications of similar queries for static data. Next, location-based term queries focus on term frequencies in streams of geo-textual objects, thus retrieving the top-k most frequent or trending terms in a region, or retrieving top-k locally frequent terms, or performing selectivity estimation.

Future work. Several areas exist where there is a substantial need for research.

Evaluation for spatial keyword queries: Many types of spatial keyword queries have been proposed to address a variety of user needs. However, the lack of ground-truth query results makes it an open problem to evaluate the effectiveness of proposed spatial keyword query functionality. An existing benchmark [32] for spatial keyword queries focus on efficiency only. There is a need for additional means that enable evaluating the effectiveness of spatial keyword queries. Such means may include evaluation procedures based on crowd-sourcing or benchmarks complete with ground-truth data and evaluation metrics. With such means in place, new and interesting research directions in spatial keyword querying will materialize, e.g., how the recent progress in deep learning for textual relevance computation can be used to improve spatial keyword querying and, e.g., enable personalized spatial keyword queries.

Querying streaming geo-textual data in road networks: So far, few studies exist on querying streaming geo-textual data in road networks. Possible applications include road traffic monitoring and on-the-ride optimal route finding. Solutions for such applications call for high efficiency in computing the spatial proximity between subscription queries and a large-scale streaming data, which is challenging. Further, studies generally do not consider travel time that varies over time as the notion of proximity, but rather consider road-network distance. In contrast, travel time is more important in many applications. Some studies beyond spatial keyword querying model travel time as time-varying distributions so that the travel time from a query to an object is a function from time to travel-time distributions. Further, depending on the application scenario spatial proximity may need to consider aspects such as road tolls and travel restrictions.

Systems to support spatial keyword queries: Most existing studies on spatial keyword queries focus on solving one or several specific query types in a stand-alone setting. While this may enable important location-based services, there are clear benefits to integrating the proposed functionality and solutions into larger systems. Thus, the development of systems that support broad ranges of spatial keyword queries as a first class citizens is an important research direction. Such systems will enable reuse of functionality and will enable applications that need to compose different types of queries. In particular, it is of great interest to extend existing relational database engines to support spatial keyword queries. This is challenging because it requires integration of spatial keyword processing techniques into query optimization and processing. As part of this, methods are needed that are capable of estimating the cardinality of operations involving spatial and textual attributes.

Machine learning for database systems supporting spatial keyword queries: It remains an open problem to build scalable systems that treat the support for spatial keyword queries as a first class citizen. It remains challenging to handle streaming geo-textual data streams at scale. Machine learning techniques have been used to solve a variety of data management problems, such as learning more compact and efficient indexes, selecting query plans more effectively, improving query optimization, or performing better data partitioning. It is of interest to explore machine learning techniques to support spatial keyword queries, particularly in the streaming setting.

Acknowledgements This research was supported in part by a MOE Tier-2 grant MOE2019-T2-2-181, a MOE Tier-1 grant RG114/19, an NTU ACE grant, and Singtel Cognitive and Artificial Intelligence Lab for Enterprises (SCALE@NTU), which is a collaboration between Singapore Telecommunications Limited (Singtel) and Nanyang Technological University (NTU) that is funded by the Singapore Government through the Industry Alignment Fund Industry Collaboration Projects Grant. The authors would like to thank the anonymous reviewers for providing constructive feedback and valuable suggestions.

References

1. 30 foursquare statistics to help you optimize the platform in 2019. `https://99firms.com/blog/foursquare-statistics/#gref`. Accessed: 2019-11-05
2. Apache lucene. `https://lucene.apache.org`. Accessed: 2019-11-05
3. Google s2 geometry library. `https://code.google.com/archive/p/s2-geometry-library/`. Accessed: 2020-11-10
4. Hdfs architecture guide. `https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html`. Accessed: 2020-11-10
5. Introduction to postgis. `https://postgis.net/workshops/postgis-intro/index.html`. Accessed: 2020-11-10
6. An introduction to yelp metrics as of 30 june, 2019. `https://www.yelp.com/factsheet`. Accessed: 2019-11-05
7. The mongodb4.4 manual mongodb manual. `https://docs.mongodb.com/manual/`. Accessed: 2020-11-10
8. Mysql 8.0 reference manual. `https://dev.mysql.com/doc/refman/8.0/en/`. Accessed: 2020-11-10
9. Oracle help center. `https://docs.oracle.com/en/`. Accessed: 2020-11-10
10. Abdelhamid, A.S., Tang, M., Aly, A.M., Mahmood, A.R., Qadah, T., Aref, W.G., Basalamah, S.: Cruncher: Distributed in-memory processing for location-based services. In: ICDE, pp. 1406–1409 (2016)
11. Abdelhaq, H., Sengstock, C., Gertz, M.: Eventweet: Online localized event detection from twitter. PVLDB **6**(12), 1326–1329 (2013)

12. Ahmed, P., Hasan, M., Kashyap, A., Hristidis, V., Tsotras, V.J.: Efficient computation of top-k frequent terms over spatio-temporal ranges. In: SIGMOD, pp. 1227–1241 (2017)
13. Ahuja, R., Armenatzoglou, N., Papadias, D., Fakas, G.J.: Geo-social keyword search. In: SSTD, pp. 431–450 (2015)
14. Almaslukh, A., Magdy, A.: Evaluating spatial-keyword queries on streaming data. In: SIGSPATIAL, pp. 209–218 (2018)
15. de Almeida, J.P.D., Rocha-Junior, J.B.: Top-k spatial keyword preference query. JIDM **6**(3), 162–177 (2015)
16. Alsubaiee, S., Behm, A., Li, C.: Supporting location-based approximate-keyword queries. In: SIGSPATIAL, pp. 61–70 (2010)
17. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D., Trevisan, L.: Counting distinct elements in a data stream. In: RANDOM, pp. 1–10. Springer (2002)
18. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE, pp. 421–430 (2001)
19. Bouros, P., Ge, S., Mamoulis, N.: Spatio-textual similarity joins. PVLDB **6**(1), 1–12 (2012)
20. Cao, X., Chen, L., Cong, G., Jensen, C.S., Qu, Q., Skovsgaard, A., Wu, D., Yiu, M.L.: Spatial keyword querying. In: ER, pp. 16–29. Springer (2012)
21. Cao, X., Chen, L., Cong, G., Xiao, X.: Keyword-aware optimal route search. PVLDB **5**(11), 1136–1147 (2012)
22. Cao, X., Cong, G., Guo, T., Jensen, C.S., Ooi, B.C.: Efficient processing of spatial group keyword queries. TODS **40**(2), 13:1–13:48 (2015)
23. Cao, X., Cong, G., Jensen, C.S.: Retrieving top-k prestige-based relevant spatial web objects. PVLDB **3**(1), 373–384 (2010)
24. Cao, X., Cong, G., Jensen, C.S., Ooi, B.C.: Collective spatial keyword querying. In: SIGMOD, pp. 373–384 (2011)
25. Cary, A., Wolfson, O., Rishe, N.: Efficient and scalable method for processing top-k spatial Boolean queries. In: SSDBM, pp. 87–95 (2010)
26. Chan, H.K., Long, C., Wong, R.C.: Inherent-cost aware collective spatial keyword queries. In: SSTD, pp. 357–375 (2017)
27. Chan, H.K.H., Long, C., Wong, R.C.W.: On generalizing collective spatial keyword queries. TKDE **30**(9), 1712–1726 (2018)
28. Chen, G., Zhao, J., Gao, Y., Chen, L., Chen, R.: Time-aware Boolean spatial keyword queries. TKDE **29**(11), 2601–2614 (2017)
29. Chen, J., Xu, J., Liu, C., Li, Z., Liu, A., Ding, Z.: Multi-objective spatial keyword query with semantics. In: DASFAA, pp. 34–48 (2017)
30. Chen, L., Cong, G., Cao, X.: An efficient query indexing mechanism for filtering geo-textual data. In: SIGMOD, pp. 749–760 (2013)
31. Chen, L., Cong, G., Cao, X., Tan, K.: Temporal spatial-keyword top-k publish/subscribe. In: ICDE, pp. 255–266 (2015)
32. Chen, L., Cong, G., Jensen, C.S., Wu, D.: Spatial keyword query processing: an experimental evaluation. PVLDB **6**(3), 217–228 (2013)
33. Chen, L., Li, Y., Xu, J., Jensen, C.S.: Direction-aware why-not spatial keyword top-k queries. In: ICDE, pp. 107–110 (2017)
34. Chen, L., Lin, X., Hu, H., Jensen, C.S., Xu, J.: Answering why-not questions on spatial keyword top-k queries. In: ICDE, pp. 279–290 (2015)
35. Chen, L., Roy, A.: Event detection from flickr data through wavelet-based spatial analysis. In: CIKM, pp. 523–532 (2009)
36. Chen, L., Shang, S., Yang, C., Li, J.: Spatial keyword search: A survey. GeoInformatica **24**(1), 85–106 (2020)
37. Chen, L., Shang, S., Zheng, K., Kalnis, P.: Cluster-based subscription matching for geo-textual data streams. In: ICDE, pp. 890–901 (2019)
38. Chen, L., Xu, J., Jensen, C.S., Li, Y.: YASK: A why-not question answering engine for spatial keyword query services. PVLDB **9**(13), 1501–1504 (2016)
39. Chen, L., Xu, J., Lin, X., Jensen, C.S., Hu, H.: Answering why-not spatial keyword top-k queries via keyword adaption. In: ICDE, pp. 697–708 (2016)
40. Chen, Y., Chen, Z., Cong, G., Mahmood, A.R., Aref, W.G.: Sstd: a distributed system on streaming spatio-textual data. PVLDB **13**(12), 2284–2296 (2020)
41. Chen, Y., Suel, T., Markowetz, A.: Efficient query processing in geographic web search engines. In: SIGMOD, pp. 277–288 (2006)
42. Chen, Z., Cong, G., Zhang, Z., Fuz, T.Z., Chen, L.: Distributed publish/subscribe query processing on the spatio-textual data stream. In: ICDE, pp. 1095–1106 (2017)
43. Choi, D.W., Pei, J., Lin, X.: Finding the minimum spatial keyword cover. In: ICDE, pp. 685–696 (2016)
44. Choudhury, F.M., Culpepper, J.S., Sellis, T.K., Cao, X.: Maximizing bichromatic reverse spatial and textual k nearest neighbor queries. PVLDB **9**(6), 456–467 (2016)
45. Christoforaki, M., He, J., Dimopoulos, C., Markowetz, A., Suel, T.: Text vs. space: efficient geo-search query processing. In: CIKM, pp. 423–432 (2011)
46. Cong, G., Feng, K., Zhao, K.: Querying and mining geo-textual data for exploration: Challenges and opportunities. In: ICDE Workshops, pp. 165–168 (2016)
47. Cong, G., Jensen, C.S.: Querying geo-textual data: Spatial keyword queries and beyond. In: SIGMOD, pp. 2207–2212. ACM (2016)
48. Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. In: PVLDB, pp. 337–348 (2009)
49. Cui, N., Li, J., Yang, X., Wang, B., Reynolds, M., Xiang, Y.: When geo-text meets security: Privacy-preserving Boolean spatial keyword queries. In: ICDE (2019)
50. Deng, K., Li, X., Lu, J., Zhou, X.: Best keyword cover search. In: ICDE, pp. 61–73 (2014)
51. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische mathematik **1**(1), 269–271 (1959)
52. Fagin, R.: Combining fuzzy information from multiple systems. In: PODS, pp. 216–226 (1996)
53. Faloutsos, C., Christodoulakis, S.: Signature files: An access method for documents and its analytical performance evaluation. TOIS **2**(4), 267–288 (1984)
54. Fan, J., Li, G., Zhou, L., Chen, S., Hu, J.: SEAL: spatio-textual similarity search. PVLDB **5**(9), 824–835 (2012)
55. Fang, H., Zhao, P., Sheng, V.S., Li, Z., Xu, J., Wu, J., Cui, Z.: Ranked reverse Boolean spatial keyword nearest neighbors search. In: WISE, pp. 92–107 (2015)
56. Fang, Y., Cheng, R., Cong, G., Mamoulis, N., Li, Y.: On spatial pattern matching. In: ICDE, pp. 293–304 (2018)
57. Feng, W., Zhang, C., Zhang, W., Han, J., Wang, J., Aggarwal, C., Huang, J.: STREAMCUBE: hierarchical spatio-temporal hashtag clustering for event exploration over the twitter stream. In: ICDE, pp. 1561–1572 (2015)
58. Gao, Y., Qin, X., Zheng, B., Chen, G.: Efficient reverse top-k Boolean spatial keyword queries on road networks. TKDE **27**(5), 1205–1218 (2015)

59. Gao, Y., Wang, Y., Yi, S.: Preference-aware top-k spatio-textual queries. In: WAIM, pp. 186–197 (2016)
60. Gao, Y., Zhao, J., Zheng, B., Chen, G.: Efficient collective spatial keyword query processing on road networks. TITS **17**(2), 469–480 (2016)
61. Göbel, R., Henrich, A., Niemann, R., Blank, D.: A hybrid index structure for geo-textual searches. In: CIKM, pp. 1625–1628 (2009)
62. Guo, L., Shao, J., Aung, H.H., Tan, K.L.: Efficient continuous top-k spatial keyword queries on road networks. In: GeoInformatica, pp. 29–60 (2015)
63. Guo, L., Zhang, D., Li, G., Tan, K., Bao, Z.: Location-aware pub/sub system: When continuous moving queries meet dynamic event streams. In: SIGMOD, pp. 843–857 (2015)
64. Guo, T., Cao, X., Cong, G.: Efficient algorithms for answering the m-closest keywords query. In: SIGMOD, pp. 405–418 (2015)
65. Güting, R.H., Valdés, F., Damiani, M.L.: Symbolic trajectories. TSAS **1**(2), 1–51 (2015)
66. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: SIGMOD, pp. 47–57 (1984)
67. Haghani, P., Michel, S., Aberer, K.: The gist of everything new: Personalized top-k processing over web 2.0 streams. In: CIKM, pp. 489–498 (2010)
68. Hariharan, R., Hore, B., Li, C., Mehrotra, S.: Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems. In: SSDBM, p. 16 (2007)
69. He, P., Xu, H., Zhao, X., Shen, Z.: Scalable collective spatial keyword query. In: ICDE Workshops, pp. 182–189 (2015)
70. Hershberger, J., Shrivastava, N., Suri, S., Tóth, C.D.: Adaptive spatial partitioning for multidimensional data streams. In: ISAAC, pp. 522–533 (2005)
71. Hoang-Vu, T.A., Vo, H.T., Freire, J.: A unified index for spatio-temporal keyword queries. In: CIKM, pp. 135–144 (2016)
72. Hu, H., Liu, Y., Li, G., Feng, J., Tan, K.: A location-aware publish/subscribe framework for parameterized spatio-textual subscriptions. In: ICDE, pp. 711–722 (2015)
73. Hu, J., Cheng, R., Wu, D., Jin, B.: Efficient top-k subscription matching for location-aware publish/subscribe. In: SSTD, pp. 333–351 (2015)
74. Huang, W., Li, G., Tan, K., Feng, J.: Efficient safe-region construction for moving top-k spatial keyword queries. In: CIKM, pp. 932–941 (2012)
75. Huiqi, H., Guoliang, L., Zhifeng, B., Jianhua, F., Yongwei, W., Zhiguo, G., Yaoqing, X.: Top-k spatio-textual similarity join. TKDE **28**(2), 551–565 (2015)
76. I.D.Felipe, V.Hristidis, N.Rishe: Keyword search on spatial databases. In: ICDE, pp. 656–665 (2008)
77. Jiang, J., Lu, H., Yang, B., Cui, B.: Finding top-k local users in geo-tagged social media data. In: ICDE, pp. 267–278 (2015)
78. Kanza, Y., Levin, R., Safra, E., Sagiv, Y.: An interactive approach to route search. In: SIGSPATIAL, pp. 408–411 (2009)
79. Kanza, Y., Safra, E., Sagiv, Y.: Route search over probabilistic geospatial data. In: SSTD, pp. 153–170 (2009)
80. Kanza, Y., Safra, E., Sagiv, Y., Doytsher, Y.: Heuristic algorithms for route-search queries over geographical data. In: SIGSPATIAL, pp. 1–10 (2008)
81. Khodaei, A., Shahabi, C., Li, C.: Hybrid indexing and seamless ranking of spatial and textual features of web documents. In: DEXA (1), pp. 450–466 (2010)
82. Krumm, J., Horvitz, E.: Eyewitness: identifying local events via space-time signals in twitter feeds. In: SIGSPATIAL, pp. 20:1–20:10 (2015)
83. Lee, T., Park, J.w., Lee, S., Hwang, S.W., Elnikety, S., He, Y.: Processing and optimizing main memory spatial-keyword queries. PVLDB **9**(3), 132–143 (2015)
84. Li, G., Feng, J., Xu, J.: DESKS: direction-aware spatial keyword search. In: ICDE, pp. 474–485 (2012)
85. Li, G., Wang, Y., Wang, T., Feng, J.: Location-aware publish/subscribe. In: KDD, pp. 802–810 (2013)
86. Li, J., Wang, H., Li, J., Gao, H.: Skyline for geo-textual data. In: GeoInformatica, pp. 453–469 (2016)
87. Li, J., Yang, Y.D., Mamoulis, N.: Optimal route queries with arbitrary order constraints. TKDE **25**(5), 1097–1110 (2013)
88. Li, M., Chen, L., Cong, G., Gu, Y., Yu, G.: Efficient processing of location-aware group preference queries. In: CIKM, pp. 559–568 (2016)
89. Li, R., Lei, K.H., Khadiwala, R., Chang, K.C.: TEDAS: A twitter-based event detection and analysis system. In: ICDE, pp. 1273–1276 (2012)
90. Li, Y., Yang, W., Dan, W., Xie, Z.: Keyword-aware dominant route search for various user preferences. In: DASFAA, pp. 207–222 (2015)
91. Li, Z., Lee, K.C.K., Zheng, B., Lee, W.C., Lee, D.L., Wang, X.: Ir-tree: An efficient index for geographic document search. TKDE **23**(4), 585–599 (2011)
92. Liu, J., Deng, K., Sun, H., Yu, G., Zhou, X., Jensen, C.S.: Clue-based spatio-textual query. PVLDB **10**(5), 529–540 (2017)
93. Liu, S., Li, G., Feng, J.: A prefix-filter based method for spatio-textual similarity join. TKDE **26**(10), 2354–2367 (2014)
94. Long, C., Wong, R.C., Wang, K., Fu, A.W.: Collective spatial keyword queries: a distance owner-driven approach. In: SIGMOD, pp. 689–700 (2013)
95. Lu, J., Lu, Y., Cong, G.: Reverse spatial and textual k nearest neighbor search. In: SIGMOD, pp. 349–360 (2011)
96. Lu, Y., Cong, G., Lu, J., Shahabi, C.: Efficient algorithms for answering reverse spatial-keyword nearest neighbor queries. In: SIGSPATIAL, pp. 82:1–82:4 (2015)
97. Lu, Y., Lu, J., Cong, G., Wu, W., Shahabi, C.: Efficient algorithms and cost models for reverse spatial-keyword k-nearest neighbor search. TODS **39**(2), 13 (2014)
98. Lu, Y., Zhang, M., Witherspoon, S., Yesha, Y., Yesha, Y., Rishe, N.: Sksopen: efficient indexing, querying, and visualization of geo-spatial big data. In: ICMLA, pp. 495–500 (2013)
99. Luo, C., Junlin, L., Li, G., Wei, W., Li, Y., Li, J.: Efficient reverse spatial and textual k nearest neighbor queries on road networks. Knowledge-Based Systems **93**, 121 – 134 (2016)
100. Luo, S., Luo, Y., Zhou, S., Cong, G., Guan, J.: Disks: a system for distributed spatial group keyword search on road networks. PVLDB **5**(12), 1966–1969 (2012)
101. Luo, S., Luo, Y., Zhou, S., Cong, G., Guan, J., Yong, Z.: Distributed spatial keyword querying on road networks. In: EDBT, pp. 235–246 (2014)
102. Ma, Y., Zhang, Y., Meng, X.: St-hbase: a scalable data management system for massive geo-tagged objects. In: WAIM, pp. 155–166 (2013)
103. Ma, Y., Zhang, Y., Meng, X.: St-hbase: A scalable data management system for massive geo-tagged objects. In: WAIM, pp. 155–166 (2013)

104. Machanavajjhala, A., Vee, E., Garofalakis, M., Shanmugasundaram, J.: Scalable ranked publish/subscribe. PVLDB **1**(1), 451–462 (2008)
105. Magdy, A., Aly, A.M., Mokbel, M.F., Elnikety, S., He, Y., Nath, S., Aref, W.G.: Geotrend: Spatial trending queries on real-time microblogs. In: SIGSPATIAL, pp. 7:1–7:10 (2016)
106. Mahmood, A., Daghistani, A., Aly, A., Tang, M., Basalamah, S., Prabhakar, S., Aref, W.: Adaptive processing of spatial-keyword data over a distributed streaming cluster. In: SIGSPATIAL, pp. 219–228 (2018)
107. Mahmood, A.R., Aly, A.M., Aref, W.G.: FAST: frequency-aware indexing for spatio-textual data streams. In: ICDE, pp. 305–316 (2018)
108. Mahmood, A.R., Aly, A.M., Qadah, T., Rezig, E.K., Daghistani, A., Madkour, A., Abdelhamid, A.S., Hassan, M.S., Aref, W.G., Basalamah, S.: Tornado: A distributed spatio-textual stream processing system. PVLDB **8**(12), 2020–2023 (2015)
109. Mahmood, A.R., Aref, W.G.: Scalable processing of spatial-keyword queries. Synthesis Lectures on Data Management **11**(1), 1–116 (2019)
110. Mahmood, A.R., Aref, W.G., Aly, A.M., Tang, M.: Atlas: on the expression of spatial-keyword group queries using extended relational constructs. In: SIGSPATIAL, pp. 45:1–45:10 (2016)
111. Mahmood, A.R., Punni, S., Aref, W.G.: Spatio-temporal access methods: a survey (2010-2017). GeoInformatica **23**(1), 1–36 (2019)
112. Meagher, D.: Geometric modeling using octree encoding. Computer Graphics and Image Processing **19**(2), 129 – 147 (1982)
113. Mehta, P., Skoutas, D., Voisard, A.: Spatio-temporal keyword queries for moving objects. In: SIGSPATIAL, pp. 55:1–55:4 (2015)
114. Memon, I., Chen, L., Majid, A., Lv, M., Hussain, I., Chen, G.: Travel recommendation using geo-tagged photos in social media for tourist. Wireless Personal Communications **80**(4), 1347–1362 (2015)
115. Nepal, S., Ramakrishna, M.: Query processing issues in image (multimedia) databases. In: ICDE, pp. 22–29. IEEE (1999)
116. Nepomnyachiy, S., Gelley, B., Jiang, W., Minkus, T.: What, where, and when: keyword search with spatio-temporal ranges. In: GIR, pp. 2:1–2:8 (2014)
117. Pat, B., Kanza, Y.: Wheres waldo? geosocial search over myriad geotagged posts. In: SIGSPATIAL, pp. 1–10 (2017)
118. Quezada, M., Araya, V.P., Poblete, B.: Location-aware model for news events in social media. In: SIGIR, pp. 935–938 (2015)
119. Ram, A., Sunita, J., Jalal, A., Manoj, K.: A density based algorithm for discovering density varied clusters in large spatial databases. IJCA **3**(6), 1–4 (2010)
120. Rao, J., Lin, J., Samet, H.: Partitioning strategies for spatio-textual similarity join. In: SIGSPATIAL, pp. 40–49 (2014)
121. Regalado, A., Goncalves, M., Abad-Mota, S.: Evaluating skyline queries on spatial web objects. In: DEXA, pp. 416–423 (2012)
122. Rocha-Junior, J.B., Gkorgkas, O., Jonassen, S., Nørvåg, K.: Efficient processing of top-k spatial keyword queries. In: SSTD, pp. 205–222 (2011)
123. Rocha-Junior, J.B., Nørvåg, K.: Top-k spatial keyword queries on road networks. In: EDBT, pp. 168–179 (2012)
124. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes twitter users: real-time event detection by social sensors. In: WWW, pp. 851–860 (2010)
125. Samet, H.: Foundations of multidimensional and metric data structures. Morgan Kaufmann series in data management systems. Academic Press (2006)
126. Sankaranarayanan, J., Samet, H., Teitler, B.E., Lieberman, M.D., Sperling, J.: Twitterstand: news in tweets. In: SIGSPATIAL, pp. 42–51 (2009)
127. Sharifzadeh, M., Kolahdouzan, M., Shahabi, C.: The optimal sequenced route query. VLDB Journal **17**(4), 765–787 (2008)
128. Shi, J., Wu, D., Mamoulis, N.: Textually relevant spatial skylines. TKDE **28**(1), 224–237 (2016)
129. Shraer, A., Gurevich, M., Fontoura, M., Josifovski, V.: Top-k publish-subscribe for social annotation of news. PVLDB **6**(6), 385–396 (2013)
130. Skovsgaard, A., Jensen, C.S.: Finding top-k relevant groups of spatial web objects. PVLDB **24**(4), 537–555 (2015)
131. Skovsgaard, A., Sidlauskas, D., Jensen, C.S.: Scalable top-k spatio-temporal term querying. In: ICDE, pp. 148–159 (2014)
132. Su, S., Zhao, S., Cheng, X., Bi, R., Cao, X., Wang, J.: Group-based collective keyword querying in road networks. Information Processing Letters **118**, 83–90 (2017)
133. Tao, Y., Sheng, C.: Fast nearest neighbor search with keywords. TKDE **26**(4), 878–888 (2014)
134. Tsatsanifos, G., Vlachou, A.: On processing top-k spatio-textual preference queries. In: EDBT, pp. 433–444 (2015)
135. Vaid, S., Jones, C.B., Joho, H., Sanderson, M.: Spatio-textual indexing for geographical search on the web. In: SSTD, pp. 218–235 (2005)
136. Wang, B., Zhu, R., Yang, X., Wang, G.: Top-k representative documents query over geo-textual data stream. WWW pp. 537–555 (2018)
137. Wang, X., Zhang, Y., Zhang, W., Lin, X.: Efficient identification of local keyword patterns in microblogging platforms. TKDE **28**(10), 2621–2634 (2016)
138. Wang, X., Zhang, Y., Zhang, W., Lin, X., Huang, Z.: SKYPE: top-k spatial-keyword publish/subscribe over sliding window. PVLDB **9**(7), 588–599 (2016)
139. Wang, X., Zhang, Y., Zhang, W., Lin, X., Wang, W.: Selectivity estimation on streaming spatio-textual data using local correlations. PVLDB **8**(2), 101–112 (2014)
140. Wang, X., Zhang, Y., Zhang, W., Lin, X., Wang, W.: Ap-tree: Efficiently support continuous spatial-keyword queries over stream. In: ICDE, pp. 1107–1118 (2015)
141. Watanabe, K., Ochi, M., Okabe, M., Onai, R.: Jasmine: a real-time local-event detection system based on geolocation information propagated to microblogs. In: CIKM, pp. 2541–2544 (2011)
142. Wu, D., Choi, B., Xu, J., Jensen, C.S.: Authentication of moving top-k spatial keyword queries. TKDE **27**(4), 922–935 (2015)
143. Wu, D., Cong, G., Jensen, C.S.: A framework for efficient spatial web object retrieval. VLDB J. **21**(6), 797–822 (2012)
144. Wu, D., Jensen, C.S.: A density-based approach to the retrieval of top-k spatial textual clusters. In: CIKM, pp. 2095–2100 (2016)
145. Wu, D., Li, Y., Choi, B., Xu, J.: Social-aware top-k spatial keyword search. In: MDM, pp. 235–244 (2014)
146. Wu, D., Yiu, M.L., Cong, G., Jensen, C.S.: Joint top-k spatial keyword query processing. TKDE **24**(10), 1889–1903 (2012)
147. Wu, D., Yiu, M.L., Jensen, C.S.: Moving spatial keyword queries: Formulation, methods, and analysis. TODS **38**(1), 7 (2013)

148. Wu, D., Yiu, M.L., Jensen, C.S., Cong, G.: Efficient continuously moving top-k spatial keyword query processing. In: ICDE, pp. 541–552 (2011)
149. Xie, X., Lin, X., Xu, J., Jensen, C.S.: Reverse keyword-based location search. In: ICDE, pp. 375–386 (2017)
150. Xu, W., Chow, C., Yiu, M.L., Li, Q., Poon, C.K.: Mobifeed: A location-aware news feed framework for moving users. GeoInformatica **19**(3), 633–669 (2015)
151. Xu, Y., Chen, L., Yao, B., Shang, S., Zhu, S., Zheng, K., Li, F.: Location-based top-k term querying over sliding window. In: WISE, pp. 299–314 (2017)
152. Yang, J., Zhang, Y., Zhou, X., Wang, J., Hu, H., Xing, C.: A hierarchical framework for top-k location-aware error-tolerant keyword search. In: ICDE, pp. 986–997 (2019)
153. Yang, M., Zheng, L., Lu, Y., Guo, M., Li, J.: Cloud-assisted spatio-textual k nearest neighbor joins in sensor networks. In: INISCom, pp. 12–17 (2015)
154. Yao, B., Tang, M., Li, F.: Multi-approximate-keyword routing in GIS data. In: SIGSPATIAL, pp. 201–210 (2011)
155. Yu, M., Li, G., Feng, J.: A cost-based method for location-aware publish/subscribe services. In: CIKM, pp. 693–702 (2015)
156. Yu, M., Li, G., Wang, T., Feng, J., Gong, Z.: Efficient filtering algorithms for location-aware publish/subscribe. TKDE **27**(4), 950–963 (2015)
157. Zhang, C., Liu, L., Lei, D., Yuan, Q., Zhuang, H., Hanratty, T., Han, J.: Triovecevent: Embedding-based online local event detection in geo-tagged tweet streams. In: KDD, pp. 595–604 (2017)
158. Zhang, C., Zhang, Y., Zhang, W., Lin, X.: Inverted linear quadtree: Efficient top k spatial keyword search. In: ICDE, pp. 901–912 (2013)
159. Zhang, C., Zhang, Y., Zhang, W., Lin, X., Cheema, M.A., Wang, X.: Diversified spatial keyword search on road networks. In: EDBT, pp. 367–378 (2014)
160. Zhang, C., Zhou, G., Yuan, Q., Zhuang, H., Zheng, Y., Kaplan, L.M., Wang, S., Han, J.: Geoburst: Real-time local event detection in geo-tagged tweet streams. In: SIGIR, pp. 513–522 (2016)
161. Zhang, D., Chee, Y.M., Mondal, A., Tung, A.K.H., Kitsuregawa, M.: Keyword search in spatial databases: Towards searching by document. In: ICDE, pp. 688–699 (2009)
162. Zhang, D., Ooi, B.C., Tung, A.K.H.: Locating mapped resources in web 2.0. In: ICDE, pp. 521–532 (2010)
163. Zhang, D., Tan, K.L., Tung, A.K.H.: Scalable top-k spatial keyword search. In: EDBT, pp. 359–370 (2013)
164. Zhang, P., Lin, H., Yao, B., Lu, D.: Level-aware collective spatial keyword queries. Information Sciences **378**, 194–214 (2017)
165. Zhang, S., Cheng, Y., Ke, D.: Event-radar: Real-time local event detection system for geo-tagged tweet streams. arXiv (2017)
166. Zhang, Y., Ma, Y., Meng, X.: Efficient spatio-textual similarity join using mapreduce. In: IAT, pp. 52–59 (2014)
167. Zhao, J., Gao, Y., Chen, G., Chen, R.: Why-not questions on top-k geo-social keyword queries in road networks. In: ICDE (2018)
168. Zhao, J., Gao, Y., Chen, G., Jensen, C.S., Chen, R., Cai, D.: Reverse top-k geo-social keyword queries in road networks. In: ICDE, pp. 387–398 (2017)
169. Zhao, K., Chen, L., Cong, G.: Topic exploration in spatio-temporal document collections. In: SIGMOD, pp. 985–998 (2016)
170. Zhao, K., Cong, G., Chin, J.Y., Wen, R.: Exploring market competition over topics in spatio-temporal document collections. VLDB Journal **28**(1), 123–145 (2019)
171. Zhao, P., Fang, H., Sheng, V.S., Li, Z., Xu, J., Wu, J., Cui, Z.: Monochromatic and bichromatic ranked reverse Boolean spatial keyword nearest neighbors search. WWW **20**(1), 39–59 (2017)
172. Zhao, S., Cheng, X., Su, S., Shuang, K.: Popularity-aware collective keyword queries in road networks. GeoInformatica **21**(3), 485–518 (2017)
173. Zheng, B., Su, H., Hua, W., Zheng, K., Zhou, X., Li, G.: Efficient clue-based route search on road networks. TKDE **29**(9), 1846–1859 (2017)
174. Zheng, B., Zheng, K., Jensen, C.S., Nguyen, Q.V.H., Su, H., Li, G., Zhou, X.: Answering why-not group spatial keyword queries. TKDE **32**(1), 26–39 (2018)
175. Zheng, B., Zheng, K., Xiao, X., Su, H., Yin, H., Zhou, X., Li, G.: Keyword-aware continuous knn query on road networks. In: ICDE, pp. 871–882 (2016)
176. Zheng, K., Su, H., Zheng, B., Shang, S., Xu, J., Liu, J., Zhou, X.: Interactive top-k spatial keyword queries. In: ICDE, pp. 423–434 (2015)
177. Zhou, Y., Xie, X., Wang, C., Gong, Y., Ma, W.Y.: Hybrid index structures for location-based web search. In: CIKM, pp. 155–162 (2005)