Finding Socio-Textual Associations Among Locations

Paras Mehta
Freie Universität Berlin
Germany
paras.mehta@fu-berlin.de

Dimitris Sacharidis
Technische Universität Wien
Austria
dimitris@ec.tuwien.ac.at

Dimitrios Skoutas
IMIS, Athena R.C.
Greece
dskoutas@imis.athena-innovation.gr

Agnès Voisard
Freie Universität Berlin
Germany
agnes.voisard@fu-berlin.de

ABSTRACT

An increasing amount of user-generated content on the Web is geotagged. This often results in the formation of user trails, e.g., sequences of photos, check-ins, or text messages, that users generate while visiting various locations. In this paper, we introduce and study the problem of identifying sets of locations that are strongly associated under social and textual criteria. We say that a location set is associated with a set of keywords if there exists a user with posts around these locations whose textual descriptions cover all keywords. We measure the strength of this association by the number of users with posts that support it. Although the problem reminisces frequent itemset mining, we show that our support measure does not satisfy the necessary anti-monotonicity property, which is used to effectively prune the search space. Nonetheless, by studying the characteristics of the support measure, we are able to devise an efficient approach. We present a basic and two optimized algorithms, exploiting an inverted or a spatio-textual index to increase efficiency. Finally, we conduct an experimental evaluation using geotagged Flickr photos in three major cities. From a qualitative perspective, the results indicate that the introduced type of query returns meaningful and interesting location sets, which are not discovered by other existing approaches. Furthermore, the proposed optimizations and the use of appropriate indexes significantly reduce computation time.

1. INTRODUCTION

With the increasingly widespread use of mobile GPS-enabled devices and social networks, the amount of geotagged content on the Web is constantly growing. A user moving around a city may upload photos, post tweets, or check in at various locations, generating a *digital trail* of activities, which can be represented as a sequence of geotagged posts. Such publicly available trails enable the analysis and extraction of location associations that are implicitly defined by the activities of city dwellers or visitors. In turn, these associations can be used to build smarter location-based services and better understand how people experience their urban environment.

In this work, we seek to find *Socio-Textual Associations* (STAs) among locations that are strongly supported by a corpus of geotagged social media content. Given a set of keywords, we say that a group of locations are socio-textually associated if a user has posts near each of these locations, and the combined keyword set of these posts contains all query keywords. The more people make an association, i.e., the stronger its support in the corpus is, the likelier it is that there exists a latent thematic connection among the locations.

Compared to previous works that search for connections among a group of locations, our work has the distinguishing and novel aspect that it considers *social and textual criteria in unison* to define associations. In one line of work (e.g., [12, 10, 15, 3, 19, 23]), which we term *Location Patterns* (LP), the objective is to determine groups, patterns or sequences of locations (or regions) that are frequent in terms of purely social criteria, i.e., how many people support them. Since the process ignores the textual aspect, the identified locations are not semantically characterized or distinguished, and thus there is no mechanism to explore or exploit the resulting groups under a thematic context. For instance, this limits queries to finding the overall most frequent sequence of locations in a given area, or the most frequent Point of Interest (POI) to visit next. Even though one could easily enrich locations with textual information *after* the mining process, say to support recommending the most frequent restaurant to visit next, the locations remain only socially associated, and not thematically, because the computed frequencies still ignore the textual aspect.

Another related line of work is the *Collective Spatial Keyword* (CSK) query [21, 4], where locations are grouped according to textual criteria (i.e., they must collectively cover the given keywords) and spatial criteria (i.e., they must be close to each other and/or to the user's location). Thus, the optimization objective is an aggregate spatial distance, instead of some evidence-based frequency metric. In other words, the strength of the association among a valid group of locations (i.e., one that covers all keywords) is defined by spatial proximity alone. Again, this *proximity-based* approach fails to establish a thematic connection evidenced by users' behavior. For example, the fact that there is a restaurant next to an art exhibition venue, does not necessarily imply that art-loving people would find this particular restaurant attractive, unless such a connection is indeed supported by a large number of posts, from the same users, containing, for example, both keywords "*art*" and "*restaurant*" around these locations. As a matter of fact, if a strong thematic association among nearby locations exists, our problem formulation will certainly capture it.

A rather straightforward way to associate locations with keywords

Table 1: Categorization of Existing Work and Ours

Line of Work	Information Exploited			Optimization Objective
	Spatial	Textual	Social	
Location Patterns (LP) [3, 10, 12, 15, 19, 23]	×		×	frequency
Collective Spatial Keyword (CSK) [4, 21]	×	×		proximity
Aggregate Popularity (AP)	×	×	×	popularity
Socio-Textual Associations (STA)	×	×	×	frequency

according to users' behavior is based on rank aggregation [8]. For each keyword, consider a ranking of locations according to the keyword popularity, i.e., the number of posts that contain it. Then, to derive a group of locations that is most associated with a set of keywords, one can simply collect the most popular location for each keyword. This approach, which we call *Aggregate Popularity* (AP), has the advantage that individual locations are strongly associated with their respective keywords, but the location set as a whole may lack a strong socio-textual association. Indeed, each location may be popular for a different type of users, hence there may be no significantly sized population for which all these locations are popular. Exactly as in the case of proximity-based associations, if a strong thematic association among popular locations exists, our socio-textual approach will discover it.

Another differentiating trait of our work is that we consider the textual information that is included in the posts themselves, and do not rely on an external categorization of locations or POIs. The reason is that we seek to exploit the *wisdom of the crowd* to also determine textual relevance, in addition to quantify the strength of derived associations. Nonetheless, our methods can be readily adapted to take into account external textual descriptions as well.

To better frame our contribution with respect to previous works, Table 1 summarizes all approaches according to the type of information they exploit, i.e., spatial, textual, or social (user id), as well as the objective they optimize for. Mining location patterns does not exploit textual information, and seeks for groups of locations that maximize the frequency with which they co-appear among users' trails. On the other hand, collective spatial keyword queries ignore the social aspect, and look for location sets that maximize their proximity (to each other and/or a target location) subject to the constraint that they cover given keywords. An approach based on aggregating popularity considers all types of information available, and strives to include locations that are individually popular for some keyword and collectively cover given keywords. Our work also considers all types of information, but optimizes for a frequency metric that counts co-appearances of locations under a certain theme/topic/context, which is defined by the given keywords.

As an example, consider a search for locations in Berlin using the keywords "*wall*", "*art*" and "*restaurant*". Figure 1 depicts the results returned by different alternative approaches for combining locations to satisfy these keywords. Our socio-textual based approach returns the following location set as the top result (star-shaped markers): ⟨ "East Side Gallery", "Hackescher Markt" ⟩. The former is a portion of the Berlin wall covered with paintings, hence hosting many posts with the keywords "*wall*" and "*art*". The latter is a popular square in the city center, hosting also a series of restaurants frequently visited by tourists and travelers. As it turns out, these locations are neither the most popular ones for each individual keyword (see locations with circle-shaped markers, returned by the AP approach) nor close to each other. Yet, they reveal an interesting association, hinting to the fact that many travelers that have visited or plan to visit the Wall, being interested in art, tend to also prefer restaurants located at Hackescher Markt.

Furthermore, a search based on CSK identified around 350 singleton locations, for which there exists at least one user with posts

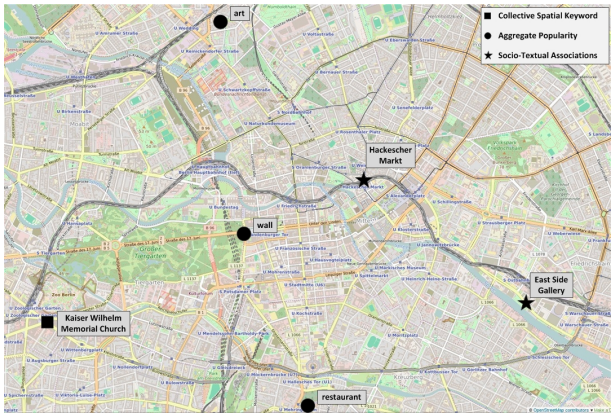


Figure 1: Example of location sets retrieved for keywords "*wall*", "*art*" and "*restaurant*" in Berlin.

containing all query keywords. One of these results is illustrated in Figure 1 (square-shaped marker). It is not straightforward how to select the best among these results; in fact, several of them may even be due to outliers or noise, which are inherent to crowdsourced content. Since a CSK query does not take frequency into account, it is better suited for cases where the query terms refer to (curated) POI categories, while being error prone and sensitive to outliers when searching on raw tags. On the other hand, the top result based on AP consists of Brandenburg Gate (for "*wall*"), a famous monument close to where the Berlin wall used to pass; the intersection of Gneisenaustr. and Mehringdamm streets (for "*restaurant*"), a place with many popular restaurants; and Stattbad Wedding (for "*art*"), a former well-known art venue. Each of these locations is popular for the respective query keyword, but they do not represent any strong shared interest between the people visiting them.

Existing algorithms for related problems cannot be used to extract socio-textual associations. Although our problem seems similar to mining frequent location patterns, the requirement for the locations to collectively cover certain keywords significantly complicates the problem, as we discuss in Section 4. Specifically, our notion of support (frequency) for a location set *does not exhibit the anti-monotonicity property* necessary to apply an Apriori-like algorithm [1]. Briefly, such a property would allow for early pruning of location sets that cannot be extended to produce valid results. Practically, the implication is that a naïve algorithm for even a relatively small-sized city-level dataset, with around 20,000 distinct locations, would need to investigate more than 10^{13} sets of three locations.

Nevertheless, by studying the problem characteristics, we are able to introduce a weaker notion of support that (1) exhibits anti-monotonicity, and (2) is an upper bound on the actual support of location sets. Armed with these two properties, we then introduce a methodology to efficiently identify location sets with strong socio-textual associations. Moreover, we study three different implementations of this methodology, each having its own merits. In the simplest, we assume that no pre-processing is allowed and that no index structure is available. We then present a method based on a simple off-the-shelf inverted index, and demonstrate how it can significantly speed up processing. The only caveat is that the association of locations with nearby posts is assumed to be known beforehand. Finally, leveraging the recent advances in spatio-textual indices, we devise an algorithm that exploits their general functionality. In particular, we consider the state-of-the-art I^3 index [22], which we also extend further to derive an even faster approach. Compared to the inverted index approach, the spatio-textual index methods allow to define the association of locations with nearby

posts dynamically, which causes an overhead in execution time but provides higher flexibility.

In addition, we consider the problem of ranking socio-textually associated location sets instead of relying on a user-specified minimum support threshold. Thus, we directly address the problem of identifying the k most strongly associated location sets. We describe a general methodology, and propose algorithms that build upon their threshold-based counterparts.

The main contributions of our work are summarized below:

- We introduce and formally define the problem of finding socio-textually associated location sets.
- We study the problem characteristics and introduce a general framework based on a weaker support measure, which satisfies the desirable anti-monotonicity property.
- We present a basic algorithm, and two efficient algorithms that exploit an inverted index and a spatio-textual index, respectively, to significantly speed up computation.
- We consider the ranking variant of the problem, and discuss the necessary adaptations to all proposed algorithms.
- We present results from an experimental evaluation using real-world data from geolocated Flickr photo trails in three major cities.

The rest of the paper is structured as follows. In the next section, we present related work. Then, we formally define the problems in Section 3, and study their characteristics in Section 4. Following this analysis, we present our algorithms in Section 5, and extend them to the top-k variant in Section 6. Finally, Section 7 presents our experimental evaluation, and Section 8 concludes the paper.

2. RELATED WORK

Next, we review related work on the topics of mining frequent locations from geotagged posts and spatial keyword search.

2.1 Mining Geotagged Posts

Several approaches analyze trails of geotagged posts, mainly photos, to extract interesting Location Patterns (LP), such as scenic routes or frequently traveled paths. A typical methodology is to use a clustering algorithm to extract landmark locations from the original posts, and then apply sequence pattern mining.

In [12], clustering is first used to identify POIs; then, association rule mining is applied to extract associative patterns among them. In [10], each photo is first assigned to a nearby POI, whereas, for the remaining ones, a density-based clustering algorithm is applied to generate additional locations. Then, a travel sequence is constructed for each user, and sequence patterns are mined from these individual travel sequences. In [15], kernel vector quantization is used to find clusters of photos; then, routes are defined as sequences of photos from the same user, and patterns are revealed by applying hierarchical clustering on routes using the Levenshtein distance. In [3], a trajectory pattern mining algorithm is applied on geotagged Flickr photos to identify frequent travel patterns and regions of interest. In [16], a clustering method is applied on geotagged photos to identify and rank popular travel landmarks.

Geotagged photos have been used to measure the attractiveness of road segments in route recommendation. A tree-based hierarchical graph is used in [24] to infer users' travel experiences and interest of a location from individual sequences. Considering the transition probability between locations, frequent travel sequences are identified. Ranking trajectory patterns mined from sequences of geotagged photos is investigated in [19]. The mean-shift algorithm extracts locations from the original GPS coordinates of the photos; then, the PrefixSpan algorithm identifies the frequent sequential patterns, which are ranked based on user and location importance. In [23], density-based clustering is used to identify regions of attractions from trails of geotagged photos; then, the Markov chain model is applied to mine transition patterns among them.

Other efforts have focused on automatic trip planning or personalized scenic route recommendations based on geotagged photo trails, taking into account user preferences, current or previous locations, and/or time budget (e.g., [13, 17]). In [6], individual photo streams are integrated into a POI graph, and itineraries are constructed based on POI popularity, available time, and destination. In [14], users' traveling preferences are learned from their travel histories in one city, and then used to recommend travel destinations and routes in a different city. In [11], a set of location sequences that match the user's preferences, present location, and time budget, are computed from individual itineraries. From a different perspective, a Bayesian approach is applied in [2] to test different hypotheses about how photo trails are produced. Various assumptions are assessed, e.g., that users tend to take photos close to the city center, near POIs, close to their previous location, or a mixture of these.

Similar to the works presented above, we also select locations that appear frequently in users' posts. However, in our case these locations should be strongly associated with a given set of keywords, a requirement which complicates the search.

2.2 Spatial Keyword Search

Spatial keyword search involves queries that comprise a user location and a set of keywords. Both the spatial and the textual parts can be applied as boolean filters or as ranking criteria. For example, the query may retrieve all relevant objects within a specified distance from the given location, or rank them based on their proximity to it; similarly, it may retrieve all objects containing one or more of the query keywords, or rank them based on relevance. A comprehensive survey of existing approaches is presented in [5].

These efforts focus on combining spatial and textual indices into hybrid ones. Accordingly, they can be characterized as text-first or space-first [7]. For example, the IF-R*-tree uses an inverted file where the postings in each inverted list are indexed by an R-tree; on the other hand, the R*-tree-IF employs an R*-tree where inverted files are attached to each leaf node [25]. More recent methods have focused on retrieving top-k objects, ranked by an aggregate score combining both spatial proximity and textual relevance [22, 20].

More closely related to our work are Collective Spatial Keyword (CSK) queries, such as the mCK query [21]. Given m keywords, it retrieves a set of spatio-textual objects that are as close to each other as possible and collectively contain all keywords. A similar variant is defined in [4], where the retrieved objects need to be as close to the user location as possible, and, optionally, in close proximity to each other.

In our work, we search for a set of locations that cover all given keywords. However, instead of optimizing for spatial proximity, we seek to maximize their co-occurrence in user trails. Thus, the approach for addressing the problem is fundamentally different.

3. PROBLEM DEFINITION

Assume a database of posts $\mathcal{P}$ made by users $\mathcal{U}$. Each post $p \in \mathcal{P}$ is a tuple $p = \langle u, \ell, \Psi \rangle$, where $p.u \in \mathcal{U}$ is the user that made the post, $p.\ell = (lon, lat)$ is the geotag (location) of the post, and $p.\Psi$ is a set of keywords that characterize it. We use $\mathcal{P}_u$ to denote all posts of user u, i.e., $\mathcal{P}_u = \{p \in \mathcal{P} : p.u = u\}$. Furthermore, assume a database of locations $\mathcal{L}$. These may correspond to the posts' locations, or, for generality, may also be defined independently of $\mathcal{P}$. For instance, one may use a POI database to populate $\mathcal{L}$, or apply a clustering algorithm on the posts' geotags and then construct $\mathcal{L}$ from the cluster centroids. Thus, we reserve the term *location* for

Table 2: Notation

Symbol	Definition
$p, \mathcal{P}$	post, database of posts
$u, \mathcal{P}_u$	user, posts of user
$\ell, L, \mathcal{L}$	location, set of locations, database of locations
ψ, Ψ	keyword, set of keywords
$\mathcal{U}_{L\Psi}$	set of users supporting (L, Ψ)
$\mathcal{U}_{L\tilde{\Psi}}$	set of users weakly supporting (L, Ψ)
$\mathcal{U}_{\Psi}$	set of users relevant to Ψ
$sup(L, \Psi)$	support of (L, Ψ)
$w\_sup(L, \Psi)$	weak support of (L, Ψ)
$rw\_sup(L, \Psi)$	relevant and weak support of (L, Ψ)
σ	support threshold

a member of $\mathcal{L}$, and refer to a post's location as its *geotag*. Table 2 summarizes the most important notation.

Locations are the principle objects in our work. We seek to identify sets of locations that are *strongly associated* with a set of keywords. To define this association, we first introduce the concepts of locality and (textual) relevance for a post.

DEFINITION 1 (LOCAL POST). *A post p is* local *to location ℓ if the post's geotag is within distance ϵ to ℓ, i.e., if $d(p.\ell, \ell) \leq \epsilon$, where d is a distance metric (e.g., Euclidean).*

DEFINITION 2 (RELEVANT POST). *A post p is* relevant *to keyword ψ if the post's keyword set contains ψ, i.e., if $\psi \in p.\Psi$.*

Posts associate locations with keywords. These associations are bestowed by users themselves, as opposed, for example, to a specific POI categorization made by a particular source; thus, they capture the wisdom of the crowd. To model the relationships between users' posts, locations, and keywords, we introduce a bipartite graph, where the two types of vertices correspond to keywords and locations, while edges correspond to users' posts.

DEFINITION 3 (ASSOCIATION GRAPH). *The* Association Graph *is a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \Psi \cup \mathcal{L}$ and $\mathcal{E} \subseteq \Psi \times \mathcal{L}$, such that an edge $e = (\psi, \ell)$ exists iff there exists at least one post p which is local to ℓ and relevant to ψ; moreover, e is labeled with the set of users that have made such posts.*

Figure 2 shows a running example with the posts of five users $u_1, \ldots, u_5$ around three locations ℓ_1, ℓ_2, ℓ_3, containing two keywords ψ_1, ψ_2. Post p_{ij} denotes the j-th post of the i-th user. For instance, post $p_{12} = \langle u_1, \ell_2, \{\psi_1, \psi_2\}\rangle$ of user u_1 is local to location ℓ_2 and relevant to keywords ψ_1 and ψ_2. The resulting Association Graph is depicted in Figure 3.

The association between a keyword and a location is explicit, and its strength can be quantified by the number of users making it. For example, three users have associated keyword ψ_1 with location ℓ_3 in the running example. On the other hand, the association between sets of keywords and sets of locations is not immediately apparent, e.g., what the textual description of the location set $\{\ell_1, \ell_2\}$ should be. If it is simply the set of keywords that have an edge towards the location set, then how do we quantify its strength if different users have made different associations? The location set should be strongly associated with a set of keywords not because there exist edges with multiple users in the Association Graph, but because there exists *a large number of users that agree on this association*. Therefore, the key question to answer is when a user supports an association between a location set and a keyword set.

DEFINITION 4 (SUPPORTING USER). *A user u* supports *the association between a location set L and keyword set Ψ, denoted as $u \in \mathcal{U}_{L\Psi}$, if:*

Users	Locations ℓ_1	ℓ_2	ℓ_3
u_1	$p_{11} : \{\psi_1\}$	$p_{12} : \{\psi_1, \psi_2\}$	$p_{13} : \{\psi_1\}$
u_2	$p_{21} : \{\psi_1\}$	$p_{22} : \{\psi_1\}$	
u_3	$p_{31} : \{\psi_2\}$	$p_{32} : \{\psi_1\}$	$p_{33} : \{\psi_1\}$
u_4		$p_{42} : \{\psi_2\}$	$p_{43} : \{\psi_1\}$
u_5	$p_{51} : \{\psi_1, \psi_2\}$		

$$L = \{\ell_1, \ell_2\}, \quad \Psi = \{\psi_1, \psi_2\}$$
$$\mathcal{U}_{L\Psi} = \{u_1, u_3\}, \quad \mathcal{U}_{L\tilde{\Psi}} = \{u_1, u_2, u_3\}$$
$$\mathcal{U}_{\Psi} = \{u_1, u_3, u_4, u_5\}, \quad \mathcal{U}_{\tilde{L}\Psi} = \{u_1, u_3, u_5\}$$
$$sup(L, \Psi) = 2, \;\; w\_sup(L, \Psi) = 3, \;\; rw\_sup(L, \Psi) = 2$$

Figure 2: Running example.

- *for each keyword $\psi \in \Psi$, the user has made a post relevant to ψ and local to a location $\ell' \in L$, i.e., every $\psi \in \Psi$ is connected via a u-labeled edge to some $\ell' \in L$; and*
- *for each location $\ell \in L$, the user has made a post local to ℓ and relevant to a keyword $\psi' \in \Psi$, i.e., every $\ell \in L$ is connected via a u-labeled edge to some $\psi' \in \Psi$.*

Hence, a user supports association (L, Ψ) if her posts connect each keyword in Ψ to some location in L, and, vice versa, each location in L to some keyword in Ψ. This implies a tight coupling between *all* keywords and *all* locations, according to the user.

An association extracted from a user's posts between a keyword set and a location set could be arbitrary. After all, the content of a post is not always related to the location where it was made, and crowdsourced content is known to be characterized by errors and noise. Hence, an association acquires credence by the number of users supporting it. Accordingly, we use this to measure the strength of a keywords-locations association.

DEFINITION 5 (SUPPORT). *The* support *of an association between a location set L and keyword set Ψ is the number of users supporting (L, Ψ), i.e., $sup(L, \Psi) = |\mathcal{U}_{L\Psi}|$.*

Returning to our example, user u_1 supports the location set $L = \{\ell_1, \ell_2\}$ and keyword set $\Psi = \{\psi_1, \psi_2\}$. For instance, post p_{11} (resp. p_{12}) is relevant to ψ_1 (resp. ψ_2) and local to some location among L; hence the first condition is satisfied; similarly, the second condition is also satisfied. It is not hard to see that the conditions are also satisfied for user u_3. Therefore, $sup(L, \Psi) = 2$.

We can now formally state the objective of this work. Given a set of keywords, we formulate two variants, one that retrieves all associations above a support threshold, and one that retrieves the k most strongly supported associations.

PROBLEM 1 (FREQUENT SOCIO-TEXTUAL ASSOCIATIONS). *Given a keyword set Ψ and a support threshold σ, identify all the location sets, up to cardinality m, that have support above σ.*

PROBLEM 2 (TOP-k SOCIO-TEXTUAL ASSOCIATIONS). *Given a keyword set Ψ, identify k location sets, up to cardinality m, that have the highest support.*

The restriction on the cardinality of the location set is because, as explained in Section 4, adding more locations can increase the support of the set.

4. OBSERVATIONS AND APPROACH

Our approach is based on some key observations regarding the intrinsic characteristics of the studied problems. In fact, the stated problems reminisce the frequent itemset problem; however, the key

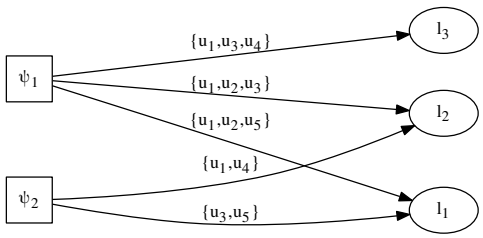


Figure 3: Association Graph for the running example.

difference here is that the introduced support function does not have the necessary anti-monotonicity property which allows for applying the Apriori principle. Given two sets X, Y, this property states that if $X \subseteq Y$, then $sup(X) \geq sup(Y)$. In other words, adding more items to a set cannot increase its support. However, the support introduced in Definition 5 does not exhibit this property.

THEOREM 1. *The support of a location set L and a keyword set Ψ is not anti-monotonic with respect to the location set, i.e., there exist two location sets $L \subseteq L'$ and a keyword set Ψ, such that $sup(L, \Psi) < sup(L', \Psi)$.*

PROOF. We prove via an example. Assume three keywords, four locations, and two users who have made posts in exactly those locations, as shown below:

	ℓ_1	ℓ_2	ℓ_3	ℓ_4
u_1	ψ_1	ψ_2	ψ_3	ψ_1
u_2	ψ_3	ψ_1	ψ_1	ψ_2

Consider the keyword set $\Psi = \{\psi_1, \psi_2, \psi_3\}$. Notice that only user u_1 supports location set $L = \{\ell_1, \ell_2, \ell_3\}$, i.e., $sup(L, \Psi) = 1$. On the other hand, both users support location set $L' = \{\ell_1, \ell_2, \ell_3, \ell_4\}$, i.e., $sup(L', \Psi) = 2$. In fact, any 3-location set in this example has support at most 1. □

As a matter of fact, the support of a location set and a keyword set can increase or decrease with respect to the location set. Despite this negative result, we devise an efficient filter-and-refine approach, where the filtering step exploits a weaker support measure.

DEFINITION 6 (WEAKLY SUPPORTING USER). *A user u* weakly supports *a given location set L and keyword set Ψ, denoted as $u \in \mathcal{U}_{L\tilde{\Psi}}$, if for each location $\ell \in L$, the user has made a post local to ℓ and relevant to a keyword in Ψ.*

The difference with respect to Definition 4 is that only the second condition applies. In other words, in the Association Graph, there must exist edges associating each one of the locations in L with keywords from Ψ, but without necessarily involving all keywords in Ψ. Accordingly, we define the notion of weak support.

DEFINITION 7 (WEAK SUPPORT). *The* weak support *of a given location set L and keyword set Ψ is the number of users weakly supporting (L, Ψ), i.e., $w\_sup(L, \Psi) = |\mathcal{U}_{L\tilde{\Psi}}|$.*

In our example, user u_2 weakly supports (L, Ψ), where $L = \{\ell_1, \ell_2\}$ and $\Psi = \{\psi_1, \psi_2\}$. For both locations, u_2 has local posts (p_{21} and p_{22}) that are relevant to at least one keyword (ψ_1). In addition, users u_1, u_3 also weakly support the same location set and keyword set. On the other hand, u_4 and u_5 do not, as they do not have posts local to both locations. Therefore, $w\_sup(L, \Psi) = 3$.

Our filter and refine approach hinges on two properties of the weak support. The first is its anti-monotonicity, while the second is that it provides an upper bound for the support of an association.

LEMMA 1. *The weak support of a location set and a keyword set is anti-monotonic with respect to the location set, i.e., for any two location sets $L' \subseteq L$ and keyword set Ψ, it holds that $w\_sup(L', \Psi) \geq w\_sup(L, \Psi)$.*

PROOF. We show that any user u that does not weakly support (L', Ψ) cannot weakly support (L, Ψ). Assume otherwise, meaning that for each location in L there exists a post of u that is local to that location and relevant to the set Ψ. Trivially, this property also holds for any location in $L' \subseteq L$. Therefore, u must also support (L', Ψ) — a contradiction. □

LEMMA 2. *The support of location set L and keyword set Ψ is not greater than their weak support, i.e., $sup(L, \Psi) \leq w\_sup(L, \Psi)$.*

PROOF. We show that any user u that supports (L, Ψ) also weakly supports (L, Ψ). As per Definition 4, u has made a post local to each location in L and relevant to a keyword in Ψ (second condition). Therefore, the condition of Definition 8 applies, and u must also weakly support (L, Ψ). □

Returning to the example, users u_1, u_2, u_3, u_5 weakly support (L', Ψ), where $L' = \{\ell_1\}$. Hence, as per Lemma 1, $w\_sup(L', \Psi) \geq w\_sup(L, \Psi)$. Moreover, as per Lemma 2, we have seen that the weak support of (L, Ψ) is one more than its support. Based on these lemmas, we can derive the following important property.

THEOREM 2. *If the weak support of a location set L and a keyword set Ψ is less than σ, then the support of any location set $L' \supseteq L$ and Ψ cannot be more than σ.*

PROOF. The premise suggests that $\sigma > w\_sup(L, \Psi)$. From Lemma 1 we have that $w\_sup(L, \Psi) \geq w\_sup(L', \Psi)$, while from Lemma 2 we get $w\_sup(L', \Psi) \geq sup(L', \Psi)$. Putting all three inequalities together we get $\sigma > sup(L', \Psi)$, i.e., the antecedent. □

This result leads us to the following filter and refine strategy. Similar to the candidate generation step of the Apriori algorithm, location sets of increasing cardinality are constructed. Then, the weak support of the set is counted, and if this is below the threshold, the set is filtered out. At the end of entire process (when set cardinality reaches m), the refinement step is perfomed by explicitly counting the support of all surviving location sets.

Still, this approach could be inefficient, producing many false positives. It is possible that the support of a location set is below the threshold even though its weak support is above the threshold. Its support may even be zero if there exists no user that has posts covering all keywords. Such a location set cannot be pruned by Theorem 2. Following our example, consider location set $L = \{\ell_1, \ell_2\}$, keyword set $\Psi = \{\psi_1, \psi_2\}$, and assume that only user u_2 exists. In this case, $w\_sup(L, \Psi) = 1$, but $sup(L, \Psi) = 0$, since there exists no post from u_2 relevant to ψ_2. Motivated by this, we seek additional ways to identify location sets that cannot have high support. We first define the notion of a relevant user.

DEFINITION 8 (RELEVANT USER). *We say that a user u is* relevant *to a given keyword set Ψ, and denote as $u \in \mathcal{U}_\Psi$, if for each keyword $\psi \in \Psi$, the user has made a post relevant to ψ, i.e., the Association Graph contains an edge that is adjacent to ψ and includes u in its label.*

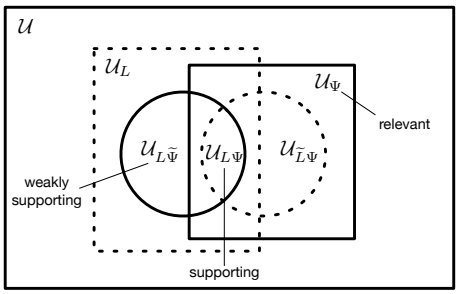


Figure 4: Set relationships between supporting, weakly supporting, and relevant users with respect to the association between location set L and keyword set Ψ.

Notice that user u_2 is not relevant to $\Psi = \{\psi_1, \psi_2\}$. The next result shows that if we restrict the set of weakly supporting users to include only relevant users, we can still define a pruning rule.

THEOREM 3. *If the number of relevant users that weakly support a location set L and a keyword set Ψ is less than σ, then the support of any location set $L' \supseteq L$ and Ψ cannot be more than σ.*

PROOF. Recall that $\mathcal{U}_\Psi, \mathcal{U}_{L\widetilde{\Psi}}$ denote the set of relevant users and weakly supporting users, respectively. Then, the theorem assumes that $|\mathcal{U}_\Psi \cap \mathcal{U}_{L\widetilde{\Psi}}| < \sigma$. From (the proof of) Lemma 1 we have that $\mathcal{U}_{L\widetilde{\Psi}} \supseteq \mathcal{U}_{L'\widetilde{\Psi}}$. Therefore, $\mathcal{U}_\Psi \cap \mathcal{U}_{L\widetilde{\Psi}} \supseteq \mathcal{U}_\Psi \cap \mathcal{U}_{L'\widetilde{\Psi}}$ and thus $|\mathcal{U}_\Psi \cap \mathcal{U}_{L\widetilde{\Psi}}| \geq |\mathcal{U}_\Psi \cap \mathcal{U}_{L'\widetilde{\Psi}}|$. From (the proof of) Lemma 2 any user u that supports (L', Ψ) must also weakly support (L', Ψ). In addition, u must be relevant to Ψ due to the first condition of Definition 4. Hence, $|\mathcal{U}_\Psi \cap \mathcal{U}_{L'\widetilde{\Psi}}| \geq sup(L', \Psi)$. Combining the two derived inequalities and the theorem assumption, we derive that $sup(L', \Psi) < \sigma$. □

This result improves upon our filter and refine strategy, by allowing us to early prune a location set that cannot have support above σ, even though its weak support might be above σ.

A better way to understand the relation between the sets of supporting $\mathcal{U}_{L\Psi}$, weakly supporting $\mathcal{U}_{L\widetilde{\Psi}}$ and relevant $\mathcal{U}_\Psi$ users of a location set and keyword set (L, Ψ) is to draw a Venn diagram. Figure 4 depicts these sets, and also includes for completeness their dual sets drawn with dashed lines (discussed in Section 5.2). We have shown that while the cardinality of set $\mathcal{U}_{L\Psi}$ is not anti-monotone with respect to L, the cardinalities of sets $\mathcal{U}_{L\widetilde{\Psi}}$ and $\mathcal{U}_\Psi \cap \mathcal{U}_{L\widetilde{\Psi}}$ are. Figure 4 emphasizes that the intersection of relevant and weakly supporting users is a *tighter* superset of the desired supporting users set, while still allowing anti-monotonicity-based prunning. In the following, we write $rw\_sup(L, \Psi)$ to denote the number of relevant and weakly supporting users, i.e., $|\mathcal{U}_\Psi \cap \mathcal{U}_{L\widetilde{\Psi}}|$.

Returning to the example of Figure 2, the relevant to Ψ users are all except u_2. Therefore, we derive $sup(L, \Psi) = |\{u_1, u_3\}| = 2$, $w\_sup(L, \Psi) = |\{u_1, u_2, u_3\}| = 3$, and $rw\_sup(L, \Psi) = |\{u_1, u_3\}| = 2$, showing that the relevant and weak support is closer to the actual support than weak support is.

5. FINDING FREQUENT ASSOCIATIONS

We first present a baseline method for Problem 1, which serves as the foundation for more elaborate solutions based on indices.

5.1 Basic Algorithm

This algorithm implements the filter and refine approach discussed in Section 4. Recall that Theorems 2 and 3 allow to prune location sets with support less than σ based on the concepts of relevant and weakly supporting users (filter step). While this guarantees no false negatives, there can still be false positives, i.e., location sets with support less than σ, which need to be identified (refine step). Note that instead of performing this at the end, it can be done more efficiently during candidate generation, as explained later.

Algorithm 1: Algorithm STA

```
   Input: keyword set Ψ, maximum cardinality m, support threshold σ
   Output: result set R^σ of all location sets with support at least σ
1  R^σ ← ∅
2  C_1 ← L                                          ▷ candidate 1-location sets
3  U_Ψ ← IdentifyRelevantUsers(Ψ)
4  for 1 ≤ i ≤ m do
5      F_i ← ∅  ▷ i-location sets with more than σ relevant and weakly supporting users
6      foreach L ∈ C_i do
7          ComputeSupports(L, Ψ)
8          if rw_sup(L, Ψ) ≥ σ then
9              F_i ← F_i ∪ {L}
10             if sup(L, Ψ) ≥ σ then
11                 R^σ ← R^σ ∪ {L}
12     C_{i+1} ← CandidateGeneration(F_i)    ▷ candidate (i+1)-location sets
```

Algorithm 2: STA.IdentifyRelevantUsers

```
   Input: keyword set Ψ
   Output: set U_Ψ of relevant users
1  U_Ψ ← ∅
2  foreach u ∈ U do
3      covΨ ← ∅
4      foreach p ∈ P_u do
5          if p.ψ ∈ Ψ then
6              covΨ ← covΨ ∪ {ψ}
7      if |covΨ| = |Ψ| then
8          U_Ψ ← U_Ψ ∪ {u}
```

Algorithm 1 outlines the basic method, denoted as STA. It operates on the set $\mathcal{P}$ of posts organized by user, i.e., the list $\mathcal{P}_u$ containing the posts of each user u. The input includes the keyword set Ψ, the maximum cardinality m of a location set, and the support threshold σ. STA exploits the Apriori principle (lines 4–12) to identify the location sets with support above σ, filtering out each location set with fewer than σ relevant and weakly supporting users.

Initially, the result set is empty and the potential 1-location sets are set to all locations (lines 1–2). Also, the set of users relevant to Ψ is identified (line 3). Procedure IDENTIFYRELEVANTUSERS, depicted in Algorithm 2, iterates across every list $\mathcal{P}_u$ and checks if user u has made posts that cover all keywords that appear in Ψ.

Then, STA proceeds in m iterations, following the Apriori principle. At the i-th iteration, all i-location sets with rw_sup not less than σ are stored in set $\mathcal{F}_i$. Among them, those with support not less than σ are added to the result set $\mathcal{R}^\sigma$. After initializing $\mathcal{F}_i$ (line 5), each candidate i-location set L is examined (lines 6–11). The set $\mathcal{C}_i$ of candidate i-location sets was generated at the end of the previous iteration (line 12) by the CANDIDATEGENERATION procedure that applies the Apriori principle. In particular, CANDIDATE GENERATION creates candidate location sets of cardinality one more than what was just examined. It takes as input the i-location sets $\mathcal{F}_i$ with relevant weak support above σ, and inserts into $\mathcal{C}_{i+1}$ an $(i+1)$-location set only if all its i-location subsets are in $\mathcal{F}_i$, due to the Apriori principle implied by Theorem 3.

For candidate i-location set L, procedure COMPUTESUPPORTS (described later) is invoked to determine the number $rw\_sup(L, \Psi)$ of relevant weakly supporting users, and the number $sup(L, \Psi)$ of supporting users (line 7). If the former support is above σ, L is added to $\mathcal{F}_i$ (lines 8–9). If, additionally, the latter support is greater than σ, then L is added to the result set $\mathcal{R}^\sigma$ (lines 10–11). This essentially corresponds to refining the surviving candidates.

Algorithm 3 depicts the pseudocode for COMPUTESUPPORTS. The

Algorithm 3: STA.COMPUTESUPPORTS

```
   Input: location set L, keyword set Ψ
   Output: weak support and support of (L, Ψ)
1  r_sup(L, Ψ) ← 0; sup(L, Ψ) ← 0
2  foreach u ∈ U_Ψ do                                   ▷ relevant user
3      covL ← ∅; covΨ ← ∅
4      foreach p ∈ P_u do
5          foreach ℓ ∈ L do
6              if d(p.ℓ, ℓ) ≤ ε then
7                  foreach ψ ∈ p.Ψ ∩ Ψ do
8                      covL ← covL ∪ {ℓ}
9                      covΨ ← covΨ ∪ {ψ}
10     if |covL| = |L| then                              ▷ weakly supporting user
11         rw_sup(L, Ψ) ← rw_sup(L, Ψ) + 1
12         if |covΨ| = |Ψ| then                          ▷ supporting user
13             sup(L, Ψ) ← sup(L, Ψ) + 1
```

Table 3: Support of Associations Between Listed Location Sets And Keyword Set $\Psi = \{\psi_1, \psi_2\}$ Based on the Posts in Figure 2

Location set	wr_sup	sup
$\{\ell_1\}$	3	1
$\{\ell_2\}$	3	1
$\{\ell_3\}$	3	0
$\boldsymbol{\{\ell_1, \ell_2\}}$	2	**2**
$\{\ell_1, \ell_3\}$	2	1
$\boldsymbol{\{\ell_2, \ell_3\}}$	3	**2**
$\{\ell_1, \ell_2, \ell_3\}$	1	1

procedure iterates over all relevant users. Let u be the currently examined user. The objective is to determine if u (weakly) supports (L, Ψ). For this purpose, the sets $covL$ and $cov\Psi$ are constructed to indicate what locations among L and keywords among Ψ, respectively, are covered by u. Each post of u is examined (lines 4–9). If the post's location is within distance ϵ to some location in $\ell \in L$, and there exists a keyword $\psi \in \Psi$ common with the post's keywords, then ℓ and ψ are inserted to $covL$ and $cov\Psi$ (lines 6–9). If all keywords in L have been found in u's relevant posts, then the counter of relevant and weakly supporting users is incremented (lines 10–11). Additionally, if all keywords appear in these posts, the counter for the support is incremented (lines 12–13).

Table 3 shows the relevant and weak support, and support for all location sets for keyword set $\Psi = \{\psi_1, \psi_2\}$, as computed by STA for the example of Figure 2 with support threshold $\sigma = 2$. Recall that all users except u_2 are relevant. As all 1-location sets have relevant and weak support above σ (although none is actually a result), all possible 2-location sets are constructed and their supports are counted. Among them, $\{\ell_1, \ell_2\}$ and $\{\ell_2, \ell_3\}$ (marked bold) have support 2 and are thus results. Observe the anti-monotonicity in relevant and weak support, and the lack thereof in support. Finally, as all 2-location sets have wr_sup above σ, the set $\{\ell_1, \ell_2, \ell_3\}$ is also considered but found to have low relevant and weak support.

5.2 Inverted Index-Based Algorithm

In STA, counting the weak support of a location set is particularly time consuming, since it scans the entire list of posts to find the weakly supporting users for each location. Even worse, if a location is part of multiple location sets, this is repeated multiple times.

To address this performance bottleneck, we present next an approach, termed STA-I, that is based on a preconstructed inverted index, which facilitates the identification of weakly supporting users for any location. The assumption here is that the distance parameter ϵ is known beforehand, i.e., it does not change with the queries.

To construct the index, we identify the posts that are within distance ϵ to each location ℓ. Then, for each location, we compile an inverted list $\mathcal{U}(\ell)$, containing all users with posts local to ℓ. To further speed up processing, we partition each list according to keyword, such that each sublist $\mathcal{U}(\ell, \psi)$ contains users with posts local to ℓ and relevant to ψ. Table 4 shows the lists for our example. STA-I operates identically to STA, but uses the inverted index during the procedures IDENTIFYRELEVANTUSERS and COMPUTESUPPORTS.

Table 4: Inverted Index for the Posts in Figure 2

Location	Inverted list
ℓ_1	$\psi_1 : u_1, u_5,\ \psi_2 : u_3, u_5$
ℓ_2	$\psi_1 : u_1, u_3,\ \psi_2 : u_1, u_4$
ℓ_3	$\psi_1 : u_1, u_3, u_4$

Algorithm 4: STA-I.IDENTIFYRELEVANTUSERS

```
   Input: keyword set Ψ
   Output: set U_Ψ of relevant users
1  U_Ψ ← ∅
2  foreach ψ ∈ Ψ do
3      C ← ∅
4      foreach ℓ ∈ L do
5          C ← C ∪ U(ℓ, ψ)
6      U_Ψ ← U_Ψ ∩ C
```

The IDENTIFYRELEVANTUSERS procedure is depicted in Algorithm 4. Recall, that the goal is to identify users who have made posts relevant to all keywords in Ψ, irrespective of the posts' geotags. Hence, for each keyword $\psi \in \Psi$, and each possible location ℓ, it retrieves the list $\mathcal{U}(\ell, \psi)$ of users with relevant and local posts, and it compiles the set of users with posts relevant to ψ and local to some location in L. Finally, it computes the intersection of these sets. This procedure essentially constructs the set of relevant users as $\mathcal{U}_\Psi = \bigcap_{\psi \in \Psi} \left(\bigcup_{\ell \in L} \mathcal{U}(\ell, \psi) \right)$.

Algorithm 5 illustrates the COMPUTESUPPORTS procedure, which computes the weak support (lines 1–6) and the support (lines 8–14) of location set and keyword set (L, Ψ). Regarding the former, recall that a user weakly supports (L, Ψ) if for each location $\ell \in L$ there exists a local post that is relevant to some keyword in Ψ. The set $\bigcup_{\psi \in \Psi} \mathcal{U}(\ell, \psi)$ represents users that have relevant posts to some keyword in Ψ and are local to the specific location ℓ. Thus the intersection over all locations in L of these sets represents the weakly supporting users, i.e., $\mathcal{U}_{L\tilde{\Psi}} = \bigcap_{\ell \in L} \left(\bigcup_{\psi \in \Psi} \mathcal{U}(\ell, \psi) \right)$. Specifically, the procedure computes the union in the inner loop (lines 3–4), and the intersection of the unions in the outer loop (lines 2–5). The weak support of (L, Ψ) is computed after the non-relevant users are discarded (line 6).

Only when the weak support of (L, Ψ) exceeds threshold σ (line 7), its support is computed (lines 8–14), but in a manner significantly different from that in STA. Recall from the discussion in Section 4 and Figure 4 that the set $\mathcal{U}_{L\tilde{\Psi}}$ of weakly supporting users has a dual set $\mathcal{U}_{\tilde{L}\Psi}$, termed local-weakly supporting users. This latter set contains users that for each keyword among Ψ have a post local to some location among L. It is not hard to see that the users that are both (relevant-)weakly supporting and local-weakly supporting (L, Ψ) are exactly those that support (L, Ψ), i.e., it holds that $\mathcal{U}_{L\Psi} = \mathcal{U}_{L\tilde{\Psi}} \cap \mathcal{U}_{\tilde{L}\Psi}$. Intuitively, the latter set satisfies the first requirement of Definition 4, whereas the former the second.

Based on this observation, the COMPUTESUPPORTS procedure first computes the local-weakly supporting users $\mathcal{U}_{\tilde{L}\Psi}$ (lines 8–13). With similar reasoning as before, the procedure builds the set as $\mathcal{U}_{\tilde{L}\Psi} = \bigcap_{\psi \in \Psi} \left(\bigcup_{\ell \in L} \mathcal{U}(\ell, \psi) \right)$, where the union is compiled in the inner loop (lines 11–12), and the intersection of the unions in the outer loop (lines 9–13). Then, it intersects it with the previously constructed $\mathcal{U}_{L\tilde{\Psi}}$ set to compute the support of (L, Ψ) (line 14).

Algorithm 5: STA-I.COMPUTESUPPORTS

Input: location set L, keyword set Ψ
Output: weak support and support of (L, Ψ)
▷ construct set $\mathcal{U}_{L\tilde{\Psi}}$ of (relevant-)weakly supporting users
1 $\mathcal{U}_{L\tilde{\Psi}} \leftarrow \emptyset$
2 **foreach** $\ell \in L$ **do**
3 $\quad \mathcal{A} \leftarrow \emptyset$ **foreach** $\psi \in \Psi$ **do**
4 $\quad\quad \mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{U}(\ell, \psi)$
5 $\quad \mathcal{U}_{L\tilde{\Psi}} \leftarrow \mathcal{U}_{L\tilde{\Psi}} \cap \mathcal{A}$
6 $rw\_sup(L, \Psi) \leftarrow |\mathcal{U}_{L\tilde{\Psi}} \cap \mathcal{U}_{\Psi}|$
7 **if** $rw\_sup(L, \Psi) < \sigma$ **then return**
▷ construct set $\mathcal{U}_{\tilde{L}\Psi}$ of local-weakly supporting users
8 $\mathcal{U}_{\tilde{L}\Psi} \leftarrow \emptyset$
9 **foreach** $\psi \in \Psi$ **do**
10 $\quad \mathcal{B} \leftarrow \emptyset$
11 $\quad$ **foreach** $\ell \in L$ **do**
12 $\quad\quad \mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{U}(\ell, \psi)$
13 $\quad \mathcal{U}_{\tilde{L}\Psi} \leftarrow \mathcal{U}_{\tilde{L}\Psi} \cap \mathcal{B}$
14 $sup(L, \Psi) \leftarrow |\mathcal{U}_{L\tilde{\Psi}} \cap \mathcal{U}_{\tilde{L}\Psi}|$

5.3 Spatio-Textual Index-Based Algorithm

Although precomputing the inverted index reduces dramatically the cost of calculating the weak support of a location set, it cannot handle different values of the distance parameter ϵ. Next, we present an alternative approach to accelerating weak support calculations based on spatio-textual indices. Instead of relying on precomputed static lists, we dynamically compile the information needed from the index. We first present a generic approach that works with the majority of existing spatio-textual indices, and then we consider a particular index and propose further optimizations.

5.3.1 Generic Algorithm

We adapt the basic Apriori-like algorithm assuming the availability of a spatio-textual index which can process spatio-textual range queries with OR semantics. The latter specify a spatial range R and a set of keywords Ψ, and seek all spatio-textual objects whose location is inside R and contain at least one of the keywords in Ψ.

We next describe the STA-ST algorithm which operates on top of such a general-purpose spatio-textual index. It operates similarly to STA, with the difference that procedure COMPUTESUPPORTS is implemented in an index-aware manner, as outlined in Algorithm 6. It first constructs the set $\mathcal{U}_{L\tilde{\Psi}}$ of weakly supporting users, and then determines the support of (L, Ψ). To build $\mathcal{U}_{L\tilde{\Psi}}$, it issues a spatio-textual range query with parameters the disc (ℓ, ϵ) of radius ϵ around each location $\ell \in L$ and keyword set Ψ (lines 2–9). For a specific location ℓ, the results (set of posts) are stored in $\mathcal{P}_\ell$ (line 4). Then, it scans the results and inserts into a temporary variable $\mathcal{A}$ each encountered user $p.u$ (line 8). In addition, it associates with each user a bitmap $p.u.cov\Psi$ indicating which query keywords appear in her posts (lines 6–7); this information is later used to determine if the user supports (L, Ψ). Once all users with posts local to ℓ and relevant to Ψ have been identified in $\mathcal{A}$, they are merged with the ones for previously examined locations (line 9). Eventually, $\mathcal{U}_{L\tilde{\Psi}}$ contains users with posts local to *every* location in L and relevant to at least one keyword in Ψ, i.e., the users weakly supporting (L, Ψ).

To compute the weak support among relevant users, the procedure takes the intersection of $\mathcal{U}_{L\tilde{\Psi}}$ with the known set $\mathcal{U}_{\Psi}$ of relevant users (line 10). If the weak support is lower than the threshold, the algorithm returns (line 11). Otherwise it computes the support by examining whether each user has covered all query keywords (lines 13–15); this is determined directly from bitmaps $p.u.cov\Psi$.

5.3.2 Optimized Algorithm

Next, we focus on a specific spatio-textual index, I^3 [22], which we adapt to devise an even more efficient algorithm.

Algorithm 6: STA-ST.COMPUTESUPPORTS

Input: location set L, keyword set Ψ
Output: weak support and support of (L, Ψ)
1 $\mathcal{U}_{L\tilde{\Psi}} \leftarrow \emptyset$
2 **foreach** $\ell \in L$ **do**
3 $\quad \mathcal{A} \leftarrow \emptyset$
4 $\quad \mathcal{P}_\ell \leftarrow$ ST-RANGE$((\ell, \epsilon), \Psi)$
5 $\quad$ **foreach** $p \in \mathcal{P}_\ell$ **do**
6 $\quad\quad$ **foreach** $\psi \in p.\Psi \cap \Psi$ **do**
7 $\quad\quad\quad p.u.cov\Psi \leftarrow p.u.cov\Psi \cup \{\psi\}$
8 $\quad\quad \mathcal{A} \leftarrow \mathcal{A} \cup p.u$
9 $\quad \mathcal{U}_{L\tilde{\Psi}} \leftarrow \mathcal{U}_{L\tilde{\Psi}} \cap \mathcal{A}$
10 $rw\_sup(L, \Psi) \leftarrow |\mathcal{U}_{L\tilde{\Psi}} \cap \mathcal{U}_{\psi}|$
11 **if** $rw\_sup(L, \Psi) < \sigma$ **then return**
12 $sup(L, \Psi) \leftarrow 0$
13 **foreach** $u \in \mathcal{U}_{L\tilde{\Psi}}$ **do**
14 $\quad$ **if** $|u.cov\Psi| = |\Psi|$ **then**
15 $\quad\quad sup(L, \Psi) \leftarrow sup(L, \Psi) + 1$

We first elaborate on the index structure. For our purposes, the I^3 index can be seen as a quadtree that hierarchically partitions the spatial domain. Each node corresponds to a specific rectangular region, and points to its four children corresponding to the quadrants of the region. Leaf nodes point to disk pages containing the actual posts grouped by keyword. We associate with each node some additional aggregate information. Specifically, for each keyword ψ, we store the number of users with relevant posts that are contained within the subtree rooted at this node N. We denote this by $N.count(\psi)$.

STA-STO differs from STA-ST in the first iteration of the main Apriori loop (lines 4–12 of Algorithm 1 for $i = 1$). Instead of computing the weak support (and support) of every location, it uses the index to identify locations with potentially high weak support, eliminating groups of locations with weak support less than σ. To achieve this, it executes a best-first search (bfs) traversal [9], performing a simple test at each node to decide whether to continue in its subtree. Intuitively, we wish to terminate bfs when no location in the subtree can have weak support greater than σ.

Let Q be the priority queue implementing bfs. For each node N entering Q, the algorithm computes $a(N) = \sum_{\psi \in \Psi} N.count(\psi)$, and uses it as the queue's priority key. At each iteration, the node N in Q with the largest $a(N)$ value is removed. If $a(N)$ is greater than or equal to σ, there may exist some location in the subtree of N with weak support greater than σ. Otherwise, a safe conclusion cannot be drawn. Hence, the algorithm calculates an additional value $b(N)$ for this node, which is an upper bound on the weak support of any location within N. Clearly, if $b(N) < \sigma$, the node contains no useful locations and can be pruned. Such pruned nodes, along with their $a()$ values, are maintained in a deleted list D, which serves in the calculation of $b()$ values as explained next. For node N, its $b(N)$ value is the sum of $a()$ values for all nodes that are in Q or in D and that are within distance ϵ to N. An important observation here is that, due to the bfs traversal and the index structure, nodes in $Q \cup D$ do not spatially overlap and hence $b(N)$ does not double count posts. To summarize, STA-STO first makes the quick $a(N) \geq \sigma$ test, and only if this fails does it compute $b(N)$ and makes the more expensive $b(N) \geq \sigma$ test. If the latter fails too, the node definitely cannot contain a location set with weak support greater than σ.

For each location dequeued in the bfs traversal, STA-STO invokes the STA-ST.COMPUTESUPPORT procedure as described in the previous section, to determine its exact weak support and its support. Compared to it, the benefit is that STA-STO executes the procedure only for promising locations instead of every possible location.

6. FINDING TOP-K ASSOCIATIONS

Next, we present algorithms for Problem 2. We start with a basic

Algorithm 7: Algorithm K-STA

Input: keyword set Ψ, maximum cardinality m, number of results k
Output: result set $\mathcal{R}^k$ containing top-k location sets with highest support

```
σ ← DetermineSupportThreshold(Ψ, k)
R^σ ← STA (Ψ, m, σ)
R^k ← k location sets from R^σ with highest support
```

approach, and then discuss more efficient index-based techniques.

6.1 Basic Algorithm

In Problem 2, we seek the top-k location sets with the highest support, instead of setting a specific support threshold. However, a support threshold is needed in order to apply an Apriori-like method; thus, we explain how such a threshold can be computed. If we pick any set of k distinct location sets and compute their supports, then the minimum value among those can serve as the support threshold σ; clearly, any other set with support lower than this cannot be in the result. The challenge is to construct initial location sets with high support so that the starting value of σ is effectively high.

Algorithm 7 outlines the generic method K-STA implementing this simple idea. First, procedure DetermineSupportThreshold is invoked to obtain an appropriate lower bound σ on the support of the top-k set. Given σ, it invokes the STA algorithm to derive all location sets with support above σ. Finally, among the returned location sets, it returns the k with the highest support.

Regarding the DetermineSupportThreshold procedure, the main idea is to construct at least k distinct location sets that cover all keywords Ψ. Suppose that for each keyword $\psi \in \Psi$ we have determined $k(\psi)$ distinct locations with local posts relevant to ψ. Combining these $k(\psi)$ distinct locations for each keyword, we can construct distinct location sets. Note that a necessary condition to obtain k location sets is $\prod_{\psi \in \Psi} k(\psi) \geq k$.

Following this process, a heuristic for obtaining combinations with high support is to start with locations that are popular, i.e., have high weak support. In the absence of any index, procedure DetermineSupportThreshold iterates over the set of posts lists $\mathcal{P}_u$, skipping users that do not have relevant posts to each ψ. For the rest, the locations of the relevant posts to each ψ are noted. In addition, a counter for the weak support of each location is maintained. After a sufficient number of locations for each keyword are seen, the procedure terminates. For each keyword, the locations with the highest weak support are chosen and combined. The support of each set is computed by ComputeSupports, and the k-th highest among these values is set as the support threshold σ.

6.2 Index-Based Algorithms

6.2.1 Inverted Index

When an inverted index from locations to users with local posts is available, DetermineSupportThreshold collects locations with local posts relevant to each keyword in Ψ in a different manner. It first computes the weak support of every location by invoking ComputeSupports. Note that this has to be executed anyway when we later invoke the STA-I algorithm irrespective of the support threshold σ. Then, it examines locations in descending order of their weak support. For each location ℓ, the procedure checks the inverted list and associates the location with each keyword in Ψ for which a local and relevant post exists. Similar to the basic algorithm, once a sufficient number of locations per keyword are seen, location sets are generated and their support is computed.

6.2.2 Spatio-Textual Index

In a generic spatio-textual index, DetermineSupportThreshold operates identically to the basic algorithm with the exception that the ComputeSupports procedure is index-aware. When the augmented I^3 index is used, a different process is followed. Procedure DetermineSupportThreshold first performs a best-first search traversal similar to that described in Section 5.3.2. The difference is that initially there is no support threshold, and thus the $b()$ values need not be computed. Moreover, the traversal is progressive, meaning that at each step the next location with potentially high weak support is identified. For each such location, its local posts are retrieved (using the index) and it is marked for the keywords that appear in these posts. As before, once a sufficient number of locations per keyword are seen, the support threshold is computed.

Table 5: Dataset Characteristics

Dataset	Num. of photos	Num. of users	Num. of distinct tags	Avg. num. of tags per photo	Avg. num. of tags per user	Num. of locations
London	1,129,927	16,171	266,495	8.1	61.2	48,547
Berlin	275,285	7,044	88,783	8.1	39.4	21,427
Paris	549,484	11,776	122,998	7.8	38.8	38,358

7. EXPERIMENTAL EVALUATION

In this section, we present an experimental evaluation of our approach using real-world datasets comprising geolocated Flickr photos. We first describe our experimental setup, outlining the datasets and the queries used in the experiments, and then we report and discuss the results.

7.1 Datasets

In our experiments, we have used geolocated photos from Flickr, extracted from a large-scale dataset that is provided publicly by Yahoo! for research purposes [18]. Specifically, we compiled datasets for the cities of London, Berlin and Paris. For each dataset, Table 5 lists the number of photos, users, and distinct keywords contained in it, as well as the average number of keywords per photo and distinct keywords per user. As a database of locations, we used POIs collected from the Foursquare API[1]. The number of distinct locations per city is also shown in Table 5.

To construct a keyword set that is used to search for socio-textual associations, we followed the process described next. First, for each dataset, we retrieved the 100 most frequent keywords, where the frequency of a keyword was measured by the number of users having photos with it. From those, we manually picked a set of 30 keywords, removing more generic ones, such as *"london"*, *"england"*, *"uk"*, *"iphone"*, *"canon"*, etc. The top 10 selected keywords for each city are listed in Table 6, showing also the number of users with relevant posts to each one. Then, we combined these popular keywords to create keyword sets of cardinality up to 4. For each case, we selected the top 20 combinations according to the number of users having photos with those tags. Table 7 lists the first 5 among these 20 combinations for each case. In all reported experiments, we set the value of the spatial distance threshold parameter ϵ, used to associate photos to locations, to 100 meters.

7.2 Indicative Result

Figure 5 shows an example of the socio-textual associations our methodology discovers. In particular, we look for locations that are strongly associated with the keyword set Ψ = {*"london eye"*, *"thames"*}. The depicted green (resp., purple) points denote the locations of photos that contain the keyword *"thames"* (resp., *"london eye"*) and belong to relevant users, i.e., they have also posted photos containing the other keyword *"london eye"* (resp., *"thames"*).

[1] https://developer.foursquare.com/

Table 6: Most Popular Keywords

London	Berlin	Paris
thames (2752)	reichstag (876)	louvre (2287)
park (1738)	fernsehturm (774)	eiffel+tower (1742)
london+eye (1730)	architecture (716)	seine (1488)
big+ben (1698)	alexanderplatz (713)	notre+dame (1244)
westminster (1543)	wall (684)	street (1194)
architecture (1519)	graffiti (575)	montmartre (1184)
museum (1386)	street (562)	architecture (1136)
art (1319)	art (543)	museum (1022)
tower+bridge (1276)	museum (526)	church (980)
statue (1178)	spree (492)	art (970)

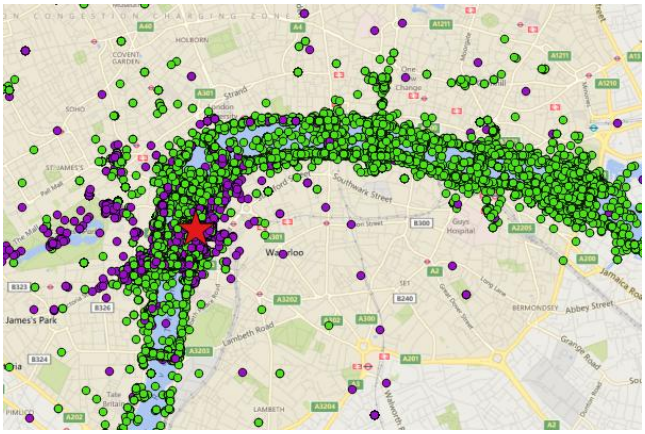


Figure 5: Indicative example for London with $\Psi = \{london\ eye, thames\}$.

We can see that photos about *"thames"* are spread along the entire length of the river Thames. On the other hand, although London Eye has a specific location, due to its high visibility relevant photos can be found at various other locations, e.g., in and around St. James Park. Nevertheless, since London Eye happens to be located at the bank of river Thames, the regions covered by the respective sets of relevant photos have a high overlap. In fact, the single location drawn as a star in this overlap has the strongest association with the keyword set. In this case, there exists a singleton location set that covers both keywords and has the highest support in the data.

7.3 Comparison with Other Association Types

As already explained (see Sections 1 and 2), there exist various approaches that discover different associations between locations and a given set of keywords. Hence, the purpose of our next experiment was to investigate whether the location sets returned by our approach (STA) are significantly different from those returned by other works, collective spatial keywords (CSK) and aggregate popularity (AP). We note that we cannot compare with approaches that discover location patterns (LP) as they ignore textual information.

To that end, we computed the top 10 results for STA, AP, and CSK, with respect to the keyword sets we compiled for the three datasets of London, Berlin, and Paris. Then, we computed the Jaccard similarity of the result sets of CSK and AP to ours. This measures the overlap in the query results, i.e., how many location sets STA and either CSK or AP return in common.

The results of this experiment are presented in Table 8. The results are averaged across queries with the same keyword set cardinality. As can be observed, the Jaccard similarity scores are very low in all cases, with values not exceeding 0.3. The highest scores are observed for queries with 2 keywords, where fewer possible location combinations exist. In those queries, on average, around

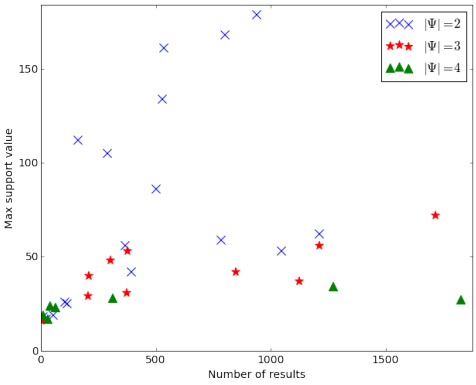


Figure 6: Scatter plot where data points correspond to experiments with distinct keyword sets; the x axis indicates the number of associations above the support threshold, and the y axis indicates the highest support among the associations.

2 or 3 of the top 10 location sets discovered by STA are common with those appearing in the results of AP or CSK. The degree of overlap drops even lower when the cardinality of the keyword set increases, allowing for a significantly larger number of candidate location sets. In those cases, often there is only one or zero results in common. This outcome is consistent across the three datasets.

These results show that STA constitutes a novel and distinct criterion for discovering interesting socio-textual associations among locations, which cannot be replicated by existing approaches.

7.4 Number of Discovered Associations and Maximum Support

Another aspect to investigate is the distribution of the number of results (associations found) and the support scores for different keyword set cardinalities. To that end, we computed the results for all keyword sets described in Section 7.1, i.e., 60 sets for each dataset, with cardinality $|\Psi| \in [2, 4]$. For each keyword set of the respective dataset, we measured the number of results and the support of the top result. The results of this experiment are shown in Figure 6. We only present results for London; the other two datasets exhibited a similar pattern, and are hence omitted. In this result, the support threshold parameter was set $\sigma = 0.1\%$ of the total number of users in the London dataset. Note that the value of the support threshold affects both the execution time and the number of results to be found. On the one hand, if the threshold is set too low, an excessive number of results may be returned, and the execution time may also be too high, since only few combinations can be pruned; on the other hand, setting the support threshold too high may return no results. Thus, the above value was selected experimentally according to this trade-off.

We notice the following trend in the results. Having only two keywords tends to produce results with high support (e.g., up to around 3% of the total number of users). As the number of keywords increases to 3 or 4, the maximum support among the returned results reduces significantly, dropping close to the support threshold; however, the number of returned results becomes much higher. This is an effect of the fact that, as explained in Section 4, the anti-monotonicity property does not hold in our problem.

7.5 Evaluation Time

Finally, we evaluate the efficiency of our proposed algorithms. In this experiment, we used the same keyword sets as above.

First, we compare the execution time of the three algorithms, `STA-I`, `STA-ST` and `STA-STO`, while varying the support threshold

Table 7: Most Popular Keyword Sets

$\|\Psi\|$	
	London
2	london+eye, thames (922); big+ben, london+eye (908); thames, westminster (898); park, thames (880); big+ben, thames (846)
3	big+ben, london+eye, thames (557); big+ben, thames, westminster (497); big+ben, london+eye, westminster (472); london+eye, thames, westminster (464); park, thames, westminster (440)
4	big+ben, london+eye, thames, westminster (358); big+ben, london+eye, thames, tower+bridge (293); art, green, park, thames (258); green, park, thames, trees (257); park, statue, thames, westminster (257)
	Berlin
2	alexanderplatz, fernsehturm (404); fernsehturm, reichstag (320); alexanderplatz, reichstag (253); reichstag, wall (249); fernsehturm, spree (248)
3	alexanderplatz, fernsehturm, reichstag (192); alexanderplatz, fernsehturm, spree (166); alexanderplatz, fernsehturm, wall (145); brandenburger+tor, fernsehturm, reichstag (144); fernsehturm, reichstag, spree (142)
4	alexanderplatz, fernsehturm, reichstag, spree (106); alexanderplatz, brandenburger+tor, fernsehturm, reichstag (96); alexanderplatz, fernsehturm, reichstag, wall (95); alexanderplatz, fernsehturm, potsdamer+platz, reichstag (90); alexanderplatz, fernsehturm, museum, reichstag (82)
	Paris
2	eiffel+tower, louvre (777); louvre, seine (745); louvre, museum (706); louvre, notre+dame (691); eiffel+tower, notre+dame (606)
3	eiffel+tower, louvre, notre+dame (415); eiffel+tower, louvre, seine (343); louvre, notre+dame, seine (339); louvre, river, seine (327); arc+de+triomphe, eiffel+tower, louvre (324)
4	eiffel+tower, louvre, notre+dame, seine (215); bridge, louvre, river, seine (209); arc+de+triomphe, eiffel+tower, louvre, notre+dame (208); louvre, museum, river, seine (189); bridge, river, seine, street (187)

Table 8: Degree of Overlap Between the Associations Discovered by STA and those of Existing Approaches

	London		Berlin		Paris	
$\|\Psi\|$	**AP**	**CSK**	**AP**	**CSK**	**AP**	**CSK**
2	0.22	0.24	0.28	0.30	0.20	0.14
3	0.17	0.04	0.09	0.07	0.08	0.03
4	0.14	0.03	0.01	0.04	0.00	0.00

Table 9: Ratio of Number of Location Sets with Support Above Threshold over Number of Location Sets with Weak Support Above Threshold; $\sigma = 0.2\%$

$\|\Psi\|$	**London**	**Berlin**	**Paris**
2	13.29%	23.80%	25.98%
3	1.35%	1.09%	3.85%
4	0.01%	0.00%	0.36%

parameter σ, which is a percentage of the number of users in each dataset. Note that the basic STA method was at least an order of magnitude slower than all other methods and is thus omitted from all plots. Moreover, we include STA-ST in the comparison, in order to assess the benefits resulting by the STA-STO optimizations. The results are presented in Figures 7 and 8, for 2 and 4 keywords, respectively; results for $|\Psi| = 3$ are similar and are omitted.

As the support threshold increases, the performance of all methods improves because fewer location sets survive the pruning. This is apparent in Paris, but not so much in London and Berlin for the specific range of support values depicted. Clearly, STA-I achieves the best performance. This is not surprising, since exploiting the preconstructed inverted index saves a substantial amount of the execution time during evaluation. It is worth noticing, however, that STA-STO is also very efficient, achieving competitive execution times compared to STA-I. In fact, this is not a merit of the spatio-textual index per se, but rather a result of the proposed optimizations; indeed, the execution times of the generic STA-ST are higher by an order of magnitude. The results appear to be consistent across the different datasets and for different number of keywords.

Table 9 quantifies the number of location sets (or associations) discovered that have weak support above but actual support below the threshold, which was set to $\sigma = 0.2\%$. For example, in London for $\Psi = 2$, we have that 13.29% of the location sets considered are actual results. As the keyword cardinality increases, the ratio decreases dramatically, because it becomes harder for location sets with weak support above the threshold to also cover all keywords.

Finally, we evaluate the performance of the algorithms for the top-k version of the problem. The results are presented in Figure 9 for $|\Psi| = 3$. A similar outcome is observed, with κ-STA-I outperforming κ-STA-STO in all cases. For both algorithms, the execution time tends to increase with k as more results are requested.

8. CONCLUSIONS

In this paper, we have addressed the problem of finding socially and textually associated location sets from user trails on the Web. We have formally defined the problem and studied its characteristics. Based on this, we have proposed a general approach for addressing the problem, which we have elaborated to derive three algorithms based on different indices. Furthermore, we have extended our approach to address also the top-k variant of the problem. The proposed methods have been evaluated experimentally using geotagged Flickr photos in three different cities.

Acknowledgements

This work was partially supported by the EU Project City.Risks (H2020-FCT-2014-653747).

9. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994.

[2] M. Becker, P. Singer, F. Lemmerich, A. Hotho, D. Helic, and M. Strohmaier. Photowalking the city: Comparing hypotheses about urban photo trails on Flickr. In *SocInfo*, pages 227–244, 2015.

[3] G. Cai, C. Hio, L. Bermingham, K. Lee, and I. Lee. Mining frequent trajectory patterns and regions-of-interest from Flickr photos. In *HICSS*, pages 1454–1463, 2014.

[4] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi. Collective spatial keyword querying. In *SIGMOD*, pages 373–384, 2011.

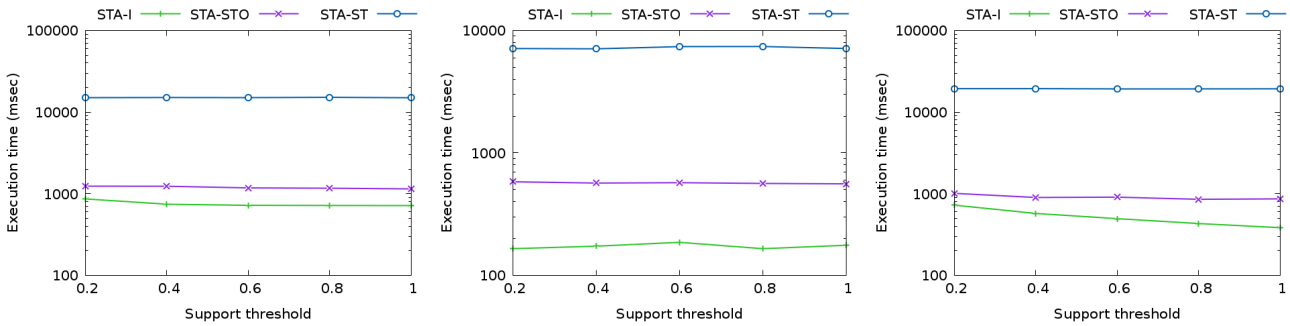
[5] L. Chen, G. Cong, C. S. Jensen, and D. Wu. Spatial keyword query processing: An experimental evaluation. *PVLDB*, 6(3):217–228, 2013.

[6] M. D. Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Automatic construction of travel itineraries using social breadcrumbs. In *HT*, pages 35–44, 2010.

[7] M. Christoforaki, J. He, C. Dimopoulos, A. Markowetz, and T. Suel. Text vs. space: efficient geo-search query processing. In *CIKM*, pages 423–432, 2011.

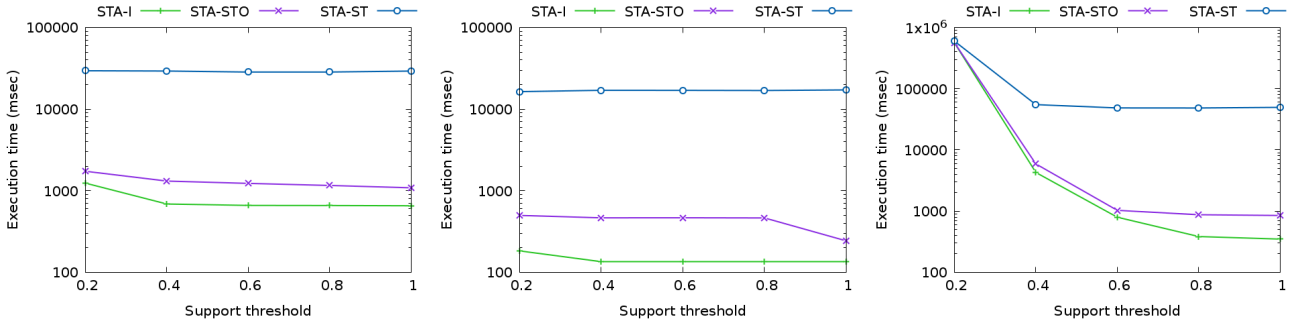
[8] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW*, pages 613–622, 2001.

[9] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM TODS*, 24(2):265–318, 1999.



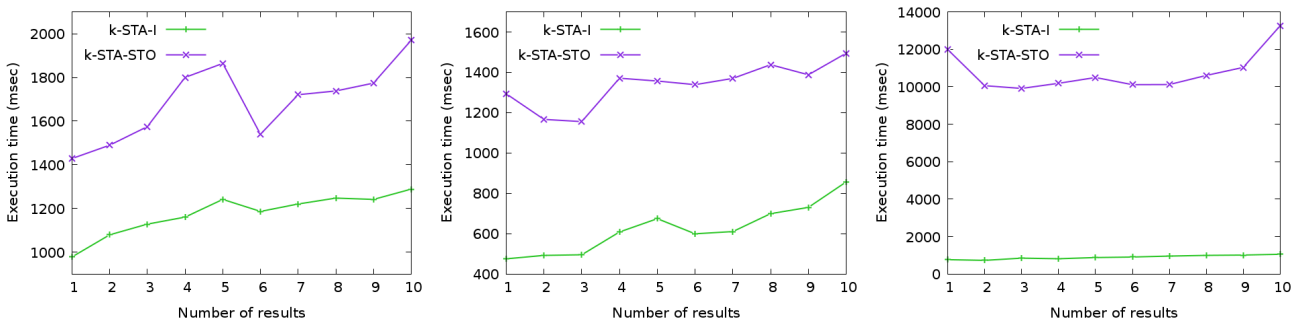
(a) London (b) Berlin (c) Paris

Figure 7: Varying support threshold (σ); $|\Psi| = 2$.



(a) London (b) Berlin (c) Paris

Figure 8: Varying support threshold (σ); $|\Psi| = 4$.



(a) London (b) Berlin (c) Paris

Figure 9: Varying number of results (k); $|\Psi| = 3$.

[10] S. Kisilevich, D. A. Keim, and L. Rokach. A novel approach to mining travel sequences using collections of geotagged photos. In *AGILE*, pages 163–182, 2010.

[11] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura. Travel route recommendation using geotags in photo sharing sites. In *CIKM*, pages 579–588, 2010.

[12] I. Lee, G. Cai, and K. Lee. Mining points-of-interest association rules from geo-tagged photos. In *HICSS*, pages 1580–1588, 2013.

[13] X. Lu, C. Wang, J. Yang, Y. Pang, and L. Zhang. Photo2Trip: generating travel routes from geo-tagged photos for trip planning. In *Multimedia*, pages 143–152, 2010.

[14] A. Majid, L. Chen, G. Chen, H. T. Mirza, I. Hussain, and J. Woodward. A context-aware personalized travel recommendation system based on geotagged social media data mining. *International Journal of Geographical Information Science*, 27(4):662–684, 2013.

[15] E. Spyrou, I. Sofianos, and P. Mylonas. Mining tourist routes from flickr photos. In *SMAP*, pages 1–5, 2015.

[16] Y. Sun, H. Fan, M. Bakillah, and A. Zipf. Road-based travel recommendation using geo-tagged images. *Computers, Environment and Urban Systems*, 53:110–122, 2015.

[17] C. Tai, D. Yang, L. Lin, and M. Chen. Recommending personalized scenic itinerary with geo-tagged photos. In *ICME*, pages 1209–1212, 2008.

[18] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. The new data and new challenges in multimedia research. *arXiv preprint arXiv:1503.01817*, 2015.

[19] Z. Yin, L. Cao, J. Han, J. Luo, and T. S. Huang. Diversified trajectory pattern ranking in geo-tagged social media. In *SDM*, pages 980–991, 2011.

[20] D. Zhang, C. Chan, and K. Tan. Processing spatial keyword query as a top-k aggregation query. In *SIGIR*, pages 355–364, 2014.

[21] D. Zhang, Y. M. Chee, A. Mondal, A. K. H. Tung, and M. Kitsuregawa. Keyword search in spatial databases: Towards searching by document. In *ICDE*, pages 688–699, 2009.

[22] D. Zhang, K.-L. Tan, and A. K. Tung. Scalable top-k spatial keyword search. In *EDBT*, pages 359–370, 2013.

[23] Y. Zheng, Z. Zha, and T. Chua. Mining travel patterns from geotagged photos. *ACM TIST*, 3(3):56, 2012.

[24] Y. Zheng, L. Zhang, X. Xie, and W. Ma. Mining interesting locations and travel sequences from GPS trajectories. In *WWW*, pages 791–800, 2009.

[25] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W. Ma. Hybrid index structures for location-based web search. In *CIKM*, pages 155–162, 2005.