ABSTRACT
The myriad geotagged posts in the social media constitute a vibrant information source that can be used to support geosocial search, that is, a search for geographic locations based on user activities in online social networks and microblogging platforms. Unlike a traditional geographic search, the results of a geosocial search are not restricted to predefined entities, and may reflect events, sentiments, and other matters that are expressed in the social media. A search for “jogging”, for instance, will indicate popular jogging places. A search for “4-th of July Fireworks” would point out places where people watch the spectacle and tweet about it. Yet, geosocial search is different from ordinary Web search because there is no natural partition of the space into documents. There is a need to find new ways to effectively rank, filter, and present results.

In this paper, we introduce a novel two-step search process of first, quickly finding relevant areas by using an arbitrarily indexed partition of the space, and second, applying clustering to the geotagged posts in the discovered areas, to present more accurate results. We propose and compare four different ranking measures for evaluating the relevance of an area to a given query. Our experiments, over a dataset of more than 40 million geotagged posts, illustrate the effectiveness of geosocial search, e.g., for finding events, or in a search based on a sentiment, in comparison to ordinary geographic search. Online search is supported by a partition-aware inverted index. Using the index, results are retrieved in a fraction of a second over millions of posts, even on a single standard machine.

CCS CONCEPTS
•Information systems →Spatial-temporal systems; Web and social media search; Social recommendation;

KEYWORDS
Geosocial search, spatio-temporal, keyword search, social media, geotagged posts, social recommendation

ACM Reference format:
DOI: 10.1145/3139958.3139962

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGSPATIAL’17, Los Angeles Area, CA, USA © 2017 ACM. 978-1-4503-5490-5/17/11 ...$15.00
DOI 10.1145/3139958.3139962

1 INTRODUCTION
The abundant geotagged posts in microblogs and online social networks, like Twitter and Instagram, are a rich source of information, where textual content is associated with location and time. While common search engines do not fully utilize such information, a geosocial search uses the geotagged social-media posts to find areas that are relevant to the given search terms. The results reflect term usage in different locations. For example, for well-known geographical entities with a distinct name, such as “High Line” (see Fig. 1) or “Bryant Park”, in New York City, many posts that contain the entity name are posted from a location that is at the entity or near it. Such searches can accurately indicate the geographical entity on a map, and they provide an evidence of the validity of the approach. There are, however, other types of search that do not refer to a specific entity. A search for “jogging”, for example, can indicate good jogging places (see Fig. 2). A search for “sunset” finds popular sunset viewpoints (see Fig. 3). A search for “Empire State Building” indicates not just the location of the building, but also popular locations from which there is a good view of the building.

Some searches are essentially abstract. For example, a search for “romantic walk” refers to how people feel in a place or how they describe it, rather than to a geographical property of the entity. In a search for “overpriced”, it is useful to see not only where this term is being used, but also other terms it is being associated with, to understand what is considered overpriced in that place.

A geosocial search may include temporal constraints, in addition to search terms. In such a case, the result is relevant areas according to posts at times that satisfy the constraints. For example, a search for “parking” with the temporal constraint “Sunday 6:00 AM – 11:00 AM” will point out locations where people tweet about parking issues on Sunday morning (on workdays, results may be different).

The examples above demonstrate that geosocial search is different from traditional geographic search. It has a merit of its own—not as a replacement of ordinary search, but as a tool for researchers, tourists and even for local residents who want to use the social media to discover places associated with given terms.

In a search it is essential to provide results instantly. This is hard in our case, since we need to process numerous geotagged posts to adequately exploit the data. To make search efficient, an index should be used. However, there are no “documents” or other natural logical atomic units to be indexed and ranked. On the one hand, without such units, we cannot build and use an inverted index. On the other hand, defining such units by partitioning the indexed area raises the concern that the partition will affect the results in an unpredicted way by returning areas that are too big or too small.

Another challenging issue is the lack of clear semantics, e.g., what should be ranked higher: a place with both many relevant posts and many irrelevant posts or a place with fewer relevant posts but where the relevant posts constitute a large percentage of the posts in this place? To properly rank the results, we need an
Figure 1: The user interface,  

Figure 2: “Jogging” places in a search for “High Line”  

Figure 3: “Sunset” viewpoints  

Figure 4: Histograms of “Lion King” vs. “Sea Lions”

effective measure of relevance. Note that the common approach of using a heat map [26] is different from ranking—it returns places that are highly active, like transport hubs, but are not necessarily relevant. Finding merely the relevant areas, over millions of posts, is often like searching for a needle in a haystack—rather than considering each post separately, we evaluate the relevance of areas, but without knowing the areas in advance.

In this paper we present a two-step approach to cope with the lack of natural partition into documents. We use an arbitrary partition of the space for indexing purposes. In a search, initially relevant areas are retrieved, using a suitable index. They serve as a first step approximation to the answer. In the second phase, clustering is applied, to find and depict fine-grained results. In order to rank the relevant partitions correctly, we propose four different models for measuring the relevance of an area.

A demonstration of geosocial search was given in [19]. However, the description of geosocial search in [19] is partial, lacking important details about the methods, their implementation, and their accuracy. The current paper elaborates on the search and how it is done. It provides an empirical evaluation of the different models and their effectiveness.

The paper has the following contributions. (1) We present geosocial search and illustrate its effectiveness in several types of search, like a search for an event, a search that relates to an abstract notion (e.g., “funny”, “romantic”), and so on. (2) We present four different search semantics and compare their effectiveness over a dataset of more than 40 million geotagged posts. (3) We compare geosocial search to ordinary geographical Web search, and describe types of queries for which geosocial search is more effective than ordinary search. (4) We show how to compute geosocial search queries using a partition-aware inverted index, and we show that using an ordinary machine and the index, relevant areas can be retrieved within a fraction of a second over dozens of millions of posts.

2 RELATED WORK

Search over social-media content receives a growing attention recently. Some systems use only the textual content of posts to find clusters of topics, e.g., Tweetmotif by O’Connor et al. [18]. Other systems also utilize post locations. For example, MapD2 and TweetMap3 provide search capabilities over geotagged tweets. However, these systems employ a “brute-force approach” and require a huge computation power while our system can run on an ordinary computer. More importantly, they provide a result similar to a heat map and do not discover and rank relevant areas. Hence, their methods are very different from ours. Thomée et al. [23] presented a generalized framework that uncovers locally characterized regions, based on geotagged data, using scale-space theory. Unlike our work the computation of the regions is offline and is limited to a pre-defined list of tags, i.e., they do not support free text search. Taghreed [16] provides the capability of querying billions of tweets, however, their focus is on finding the relevant tweets rather than ranking relevant places. STEWARD [15] finds references to geographical locations in unstructured textual documents.

Utilizing or analyzing spatio-temporal data from the social media is an active research area. TwitterStand was developed to provide local news based on tweets [11, 21]. Ferrari et al. [7] analyzed Twitter posts and applied LDA to the data, to extract urban patterns such as hotspots and crowded behavior. Xia et al. [27] showed how to detect in real time hotspots and hyper-local events in a city. Kling et al. [14] showed how to classify posts into urban topics related to various activities. Ahern et al. [1] analyzed textual tags of georeferenced photos and showed how to visualize representative tags on a map. The Livehoods project [3] used geotagged posts to find boundaries between neighborhoods. Noulas et al. [17] used Foursquare information to model crowd activity patterns in London and New York City using spectral clustering. Deng et al. [5] used DBSCAN and co-occurrence analysis to extract geospatial concepts out of social media data. Intagorn [10] proposed a probabilistic approach for mining geographical data based on social annotations. They focused on three issues: extracting the semantic of places, predicting the location of photos, and linking places to one another. Tamura et al. [22] propose a spatial clustering algorithm for extracting local regions in georeferenced documents, by finding spatially and semantically separated spatial clusters. Vaca et al. [24] introduced a taxonomy-driven framework for discovering functional areas in a city. They presented an algorithm that clusters points based on density and semantic similarity. Geosocial similarity based on sets of locations that are extracted from geotagged posts was studied

1See https://www.youtube.com/watch?v=LZjelH3ih90
2http://mapd.csail.mit.edu
3http://worldmap.harvard.edu/tweetmap/
in [13]. The discovery of a region from a set of points was studied by Duckham et al. [6].

In comparison, we show how to conduct efficient online free-text geosocial search, combined with temporal constraints, where actual relevant locations (not just a heat map) are returned. We introduce new semantics for geosocial search, show how to utilize an inverted index to compute them, and introduce specialized methods to depict the results. This provides novel geosocial search capabilities.

3 GEOSOCIAL SEARCH
We define now our framework, explain what geosocial search is, and present our relevance models.

3.1 Framework
Geosocial search is a free text search for geographic regions, based on geotagged social-media posts. Each geotagged post is a 3-tuple \( p = (c, l, t) \), where \( c \) is the textual content, \( l \) is the location where the post was created, delineated by longitude and latitude, and \( t \) is the posting time.

A geotagged dataset \( D \) is a collection of geotagged posts bounded to a particular geographic region (e.g., New York City). A search query consists of (1) a set of keywords, e.g., “Thanksgiving parade”, and (2) a temporal constrain, e.g., “Monday 8:00-9:00 AM”. Each search is executed over a certain regional boundary, e.g., the New York City area. We denote a search query as \( q = (K, \tau) \), where \( K \) is a set of keywords (terms) and \( \tau \) is a temporal constraint.

When a query is posed, the search engine (SE) tries to return the best matched regions for the user intent behind the query. For example, when the user poses the query “romantic walk”, regions that capture the user intent could include areas which are suitable for a romantic walk, a known route for a romantic walk, or maybe a place where the romantic movie “A walk to remember” was filmed. The user could specify a temporal constraint, e.g., the search for “romantic walk” with the constraint “on Sunday”, in which case the SE should retrieve regions that are relevant to Sunday.

Retrieved regions are generated \( ad \) \( hoc \), per query, and are not defined in advance. We refer to those regions as relevant regions. In a real search, there is no ground truth, however, in experiments to evaluate the accuracy of search methods, a ground truth is given in the form of an annotated map. The precision is the fraction of retrieved areas that intersect the annotated areas. Definition 6.1 specifies this formally. The goal is to rank areas such that for a given \( k \), the precision of the top-\( k \) results will be as high as possible.

Geosocial search relies on the connection between the spatiotemporal and the textual characteristics of posts, e.g., see [8]. In other words, when a post is generated, its location and time often affect its textual content. The underlying premise is that a place in which there are many relevant posts is a relevant result. When a temporal constraint is specified in the query, the query is applied to the posts whose posting time satisfies the temporal constraint. Also in this case, a place is considered relevant when there are many relevant posts associated with it, based on the geotag. As more relevant posts are generated in an area, the relevance of the area increases.

We consider a post \( p = (c, l, t) \) as relevant for a query \( q = (K, \tau) \) if the textual content of the post contains the keywords (i.e., \( K \subseteq c \)), and the time of the post, \( t \), satisfies the time constraint \( \tau \). Note that we use here a binary model of relevancy of posts rather than a model like TF-IDF because our goal is not to rank posts based on their relevance, it is to rank areas based on the posts they contain. Relevance models are discussed in Section 3.2.

The relevant areas are depicted on a map and the user should be able to understand what makes them relevant to the search. Unlike a traditional Web search, in which the resulting documents are self-explanatory, in our case, the resulting regions are assembled on the fly and the reason for their relevance is not always obvious. We implemented two solutions to overcome this gap. First, when users click on a presented region, they can view a histogram of the most frequent terms in the relevant posts of the region, e.g., Fig. 4 presents the difference between Broadway (where the “Lion King” musical is presented) and Central Park Zoo (where there are sea lions), in a search for “Lion”. For Broadway, the histogram contains the terms “lion”, “king”, “Broadway”, “show”, whereas for the zoo, the terms “lion”, “sea”, “zoo”, and “park” are shown. Second, presenting a sample of the underlying relevant posts provides information about the significance and relevance of the region.

3.2 Relevance Models
Geosocial search uses the connection between what people communicate in their posts and where they share that. As explained before, the search infers relevant areas by aggregating relevant posts. We introduce three different models to measure the relevance of an area: Global Ratio, Local Ratio, Harmonic Mean; and we compare them to Binomial Test—a common non-parametric statistical test.

Before describing each measure, we introduce some notations. We denote by \( A \) a polygonal area in the search region. By \( \text{posts}(A) \) we denote the set of posts in the area \( A \). We denote by \( \text{relevant}(A, q) \) the set of relevant posts in the spatial area \( A \), for a given query \( q \). Note that \( \text{relevant}(A, q) \subseteq \text{posts}(A) \). We denote by \( \text{relevant}(D, q) \) the set of all relevant posts in the dataset \( D \). For any set of posts \( P \), we denote by \( |P| \) the number of posts in \( P \).

There are different ways to assign a relevance score to an area, according to the type of search one would like to conduct. Next, we present the relevance measures that we study in this paper. A synthetic example to illustrate the differences between the measures is depicted in Fig. 5.

Global Ratio. Global Ratio (GR) is built on the premise that relevant areas contain a high number of relevant posts. For a given area \( A \), Global Ratio is the ratio of the relevant posts in \( A \) to the relevant posts in the entire dataset \( D \). If there are no relevant posts in \( D \), then \( GR(A, q) = 0 \). Otherwise,

\[
GR(A, q) = \frac{|\text{relevant}(A, q)|}{|\text{relevant}(D, q)|}
\]

In Fig. 5, the number of relevant posts is 10. Hence, Cell 1, Cell 3, and Cell 8 have a Global Ratio score of \( \frac{10}{10} = 1 \), \( \frac{10}{10} = 1 \), and \( \frac{6}{10} = 0.6 \), respectively. Cell 8 has the highest GR score.

Global Ratio is effective in finding places when particular words are associated with them. For example, in Manhattan the terms “Bryant Park” distinctively point out the location of the park because many of the posts in NYC with these terms are created in the park. The weakness of Global Ratio is that it prefers places with high activity to places with low activity. For example, general terms like “New York” or “happy” will receive a high Global Ratio score.
In places like Times Square or Penn Station simply because many posts are created in these places. To cope with that, we present additional relevance models.

**Local Ratio.** While Global Ratio measures the number of relevant posts in the examined area, Local Ratio (LR) measures the fraction of the relevant posts in the area. For a given area $A$, the LR score is the ratio of the number of relevant posts in $A$ to the total number of posts in $A$. If there are no posts in $A$ (i.e., $\text{posts}(A) = 0$), then $LR(A, q) = 0$. Otherwise,

$$LR(A, q) = \frac{\text{Relevant posts in } A}{\text{All posts in } A} = \frac{|\text{relevant}(A, q)|}{|\text{posts}(A)|}.$$

In Fig. 5, the LR scores for Cell 1, Cell 3, and Cell 8 are $\frac{3}{5} = 0.33$, $\frac{1}{2} = 0.5$, and $\frac{6}{32} = 0.18$, respectively. Cell 3 receives the highest LR score because a large percentage of its posts are relevant. In Cell 8, where the GR score is the highest, the percentage of relevant posts is not as high as in Cell 3.

The Local Ratio model is effective in pointing out places where a large percentage of the posts in them are relevant to the query. However, this model often gives a high relevance score to places with low networking activity, because in such places each relevant post has a higher relative weight.

**Harmonic Mean.** Both GR and LR have advantages and drawbacks. Global Ratio measures the relevance on a global scale, hence having more posts is better. Consequently, it favors bigger areas or areas with a high concentration of posts. It frequently chooses the most active areas on the map, e.g., Times Square in NYC. Contrarily, Local Ratio measures the local relevance, therefore it favors smaller areas with a low density of posts. This might increase the probability of false classification when even a small number of posts can make those areas relevant.

We want to balance these two contradicting factors. There are many ways to balance two scores, and our main concern is to eliminate noise in the two measures, e.g., by avoiding giving a high score to areas that got a very high score in one of the measures but a low score in the other. We balance the two scores by calculating their harmonic mean (HM). Only areas that receive high scores for both LR and GR have a high harmonic mean score.

For a given area $A$, if $LR(A, q) = 0$ or $GR(A, q) = 0$ then the score is defined to be 0. Otherwise, the score is the harmonic mean of the local ratio and the global ratio scores.

$$HM(A, q) = \frac{2}{\frac{1}{LR(A, q)} + \frac{1}{GR(A, q)}}$$

In Fig. 5, the Harmonic Mean scores for Cell 1, Cell 3, and Cell 8 are $\frac{2}{\frac{1}{0.33} + \frac{1}{0.5}} = 0.31$, $\frac{2}{\frac{1}{0.5} + \frac{1}{0.18}} = 0.16$, and $\frac{2}{\frac{1}{0.18} + \frac{1}{0.05}} = 0.27$, respectively. The score given to Cell 3 is much lower than the scores of Cell 1 and Cell 8. This illustrates how HM eliminates noisy scores—even though Cell 3 has a high LR score (0.5), there are not enough relevant posts to support that, hence the low GR score. Also a place with a high GR score but a low LR score will have a low HM score.

**Binomial Test.** Instead of accumulating and counting relevant posts, it is possible to apply a non-parametric statistical test. For a given area $A$ we consider each post to be an independent Bernoulli trial. A relevant post is considered as a success and an irrelevant one, as a failure. This yields a Bernoulli process where the number of successes has a binomial distribution.

The probability of a success event is derived from the total number of relevant posts in the dataset. That is, given a dataset $D$ and a query $q$, we consider the probability of success in each Bernoulli trial to be $p_{\text{success}} = \frac{|\text{relevant}(D, q)|}{|D|}$. For an examined area $A$ with $n$ observations (i.e., $n = |\text{posts}(A)|$), the Bernoulli test is applied $n$ times. We calculate the $p$-value likelihood of seeing at least $k = |\text{relevant}(A, q)|$ success results (relevant posts) in $A$, using the binomial test. ‘The smaller the $p$-value likelihood is, the smaller the probability that area $A$ came from the binomial distribution is. That is, the region is more unique, so the region score should be higher. Hence, the score is $1 - p$-value.

**Example 3.1.** Consider a dataset $D$ with 1000 posts, and a query $q$ such that 50 posts of $D$ are relevant. The success probability is $p_{\text{success}} = 0.05$. Suppose that area $A_1$ has 100 posts and 5 of them are relevant. The probability of 5 or more successes in such a case is $P(X \geq k) \approx 0.564$, where $X$ is the the number of successes. So, $A_1$ will receive a score of $1 - 0.564 = 0.436$. Now consider an area $A_2$ also with 100 posts, but where 10 of them are relevant. Now, $P(X \geq 10) \approx 0.028$, so the score given to $A_2$ is 0.972. An area with 10 posts that 5 of them are relevant will receive a score that is very close to 1 (approximately 0.99994).

Let $n = |\text{posts}(A)|$ be the number of posts in the tested area $A$ (the number of trials). Let $k = |\text{relevant}(A, q)|$ be the number of relevant posts in $A$ (the number of successes). Let $p_{\text{success}} = \frac{|\text{relevant}(D, q)|}{|D|}$ be the probability of a post to be relevant (the probability of a success event). The binomial test returns the $p$-value of seeing $k$ or more
Where’s Waldo? Geosocial Search over Myriad Geotagged Posts

be a need to retrieve all the relevant posts for each cell, test their relevance and compute the score of each cell. This would require parsing all the posts per each query, to test the relevance of posts. For the dataset and the machine we used in our experiments, it would take hours to compute a query.

In Web search, relevant documents are retrieved using an inverted index. For each word, the index has an entry with a posting list of all the relevant documents, sorted by their relevance score. But retrieving all the relevant posts using the index would yield a set of posts, rather than locations. Separating the retrieved posts into cells, in real time, is too expensive for online search. To cope with that, we use a partition-aware inverted index, similar to the index that was proposed in [12, 25]. This hybrid approach combines an inverted index with the spatial index. Differently from [12, 25], we use the index to find the number of relevant posts in each cell.

4.1 Partition-Aware Inverted Index

The search boundary is divided into sections. The sections are numbered. In our implementation, we used a partition into cells, by a grid, however, other partitions may be used. Suppose that there are \( n \) cells. The bottom left cell is given the number 1 and the top right cell is Cell \( n \). We denote Cell \( i \) by \( c_i \), and its posts by \( P_i \). The posts in each set \( P_i \) are numbered. The id of a post is the concatenation of the cell number and the post number. That is, post \( j \) of Cell \( i \) is denoted \( p_{i,j} \) and its id is the pair \( i \) and \( j \).

In the partition-aware inverted index, the posting lists are sorted by post id. So, the posts of each cell are clustered. In addition, the index contains for each cell the number of posts in that cell. (This is needed for the computation of the \( LR, HM \) and \( BT \) measures.)

4.2 Retrieving Relevant Cells

When computing a query, the first step is to retrieve the relevant cells, using the index. By relevant cells, we refer to cells that overlap the areas of the result. The query answer is computed in the second step, based on the relevant cells.

Example 4.1. Consider a search over a dataset \( D \) of 180 posts, where the area of the search is partitioned into 9 cells, and each cell contains 20 posts. Let \( q \) be a query with a search term \( t_1 \). Suppose that there are 10 relevant posts in \( D \), and their ids are \{11, 12, 31, 51, 52, 53, 81, 82, 83, 91\}. Consider computation of the scores for Cell 1. Using the posting list of \( t_1 \), as depicted in Fig. 6, the relevant posts can be retrieved without scanning the entire list. For Cell 1, the following scores are computed: \( GR(\text{Cell } 1, q) = \frac{4}{10} = 0.2; \)
\( LR(\text{Cell } 1, q) = \frac{4}{180} = 0.1; \)
\( HM(\text{Cell } 1, q) = \frac{4}{25} = \frac{2}{15} \).

For a query that contains several terms, \( t_1, \ldots, t_m \), the list \( L \) of posts that contain all the terms is computed by intersecting the \( m \) sorted posting lists \( l_1, \ldots, l_m \) of \( t_1, \ldots, t_m \). The computation is by finding the lists \( l_1, \ldots, l_m \) in the index, and scanning them iteratively—for each list, a pointer to its first element is initiated, and in each iteration the pointer(s) pointing to the smallest id is (are) advanced. Each time that all the \( m \) pointers point to the same post id, this post is added to the result.

To effectively scan posting lists, we use links similar to skip lists [20]. Each set of post ids of the same cell are considered a cluster. There is an anchor between every two adjacent clusters, and there
is a link from the anchor to the following anchor. This allows “jumping” from the cluster of Cell $i$ to the beginning of the cluster of the next non-empty cell. In Fig. 6, the anchors are the thick black lines labeled idx1, idx2, idx3, and idx4. Using the skip lists, $L$ is computed in sublinear time in the size of the posting lists [20].

4.3 Count Posts per Cell

Given list $L$, the number of relevant posts in each cell is counted. List $L$ is sorted by post id, and is therefore clustered (grouped) by cells. Its length is the number of relevant posts in total.

A simple way to count relevant posts per cell is by using Linear-Count (LC). Algorithm LC scans $L$ and counts the relevant posts in each cluster (posts of a single cell). This is done in linear time—$O(n)$, where $n$ is the number of posts in $L$—and in $O(k)$ memory footprint, where $k$ is the number of cells containing relevant posts.

We implemented two efficient methods to count the number of relevant posts in each cell, namely Binary-Count (BC) and Improved-Binary-Count (IBC). BC and IBC conduct the count by finding where clusters end in a binary-search fashion.

Algorithm BC iterates through $L$ and finds the places where cells are separated, using a binary search. In each step, initially BC points to the first post of Cell $j$ (say $p_{ij}$), then a binary search over all the posts of $L$ that follow $p_{ij}$ is executed, to find the last post of Cell $j$. The time complexity of each step is $O(\log n)$, for a list $L$ of size $n$. For $k$ relevant cells, there are $k$ steps, thus the total time complexity is $O(k \cdot \log n)$, with $O(k)$ memory footprint.

Algorithm IBC is an improvement of BC. It also iterates through $L$ and finds the anchors one by one. However, additional data is accumulated in each step, and every item that is read is used to shrink a future search range. The accumulated data is stored in an array $C_{ids}[k][2]$, with $k$ columns—one for each cell, and two rows. IBC stores in $C_{ids}[i][1]$ the location in $L$ of the first post of Cell $i$, and in $C_{ids}[i][2]$ it stores the location of the last post of Cell $i$.

The code of IBC is presented as Alg. 1. In line 4, $c_{current}$ is set to point to the first relevant cell (the first cell that contains relevant posts). The loop in line 7 iterates through the cells, and for each cell, the loop in line 11 applies a binary search for the end of the cell’s posts. When the end of the cluster is discovered (line 14), it is updated in array $C_{idx}$. Otherwise, the search continues, while updating $C_{idx}$ to decrease the search range in the following iterations.

**Proposition 4.2.** Algorithm IBC has $O(k \cdot \log \frac{n}{k})$ time complexity and $O(k)$ memory footprint.

**Proof.** There are $k$ columns in $C_{idx}$ and a fixed number of rows. Thus, the memory footprint is $O(k)$. To analyze the time complexity, note that the searches are in a decreasing range. The first search is over all the $n$ posts, and takes $O(\log n)$ time. It produces unexamined segments of size $\frac{n}{2}$, $\frac{n}{4}$, ... The search over the $\frac{n}{2}$ unexplored segment produces segments of size $\frac{n}{4}$, $\frac{n}{8}$, ... Overall, there are $1$ search over a segment of size $\frac{n}{2}$, $2$ searches over segments of size $\frac{n}{4}$, $4$ searches over segments of size $\frac{n}{8}$, and so on. The number of searches is $k - 1$ because there are $k$ cells (the end
of the last cluster is \( p_n \), so it is known). Hence the time complexity is \( O(n + \sum_{i=1}^{k} 2^{-i} \log \frac{k}{2^n}) = O(n + \sum_{i=1}^{k} 2^{-i} \log (n - \log 2^n)) = O(n + \sum_{i=1}^{k} 2^{-i} \log n - \log 2^n) = O(n + \log n + \sum_{i=1}^{k} 2^{-i} \log 2^n) = O(n + \log n - \sum_{i=1}^{k} 2^{-i} \log 2^n - 1) = O(k \log n - k \log k) = O(k \log \frac{n}{k}). \)

Algorithms LS, BC and IBC return the number of relevant posts in each cell, to support computing the cell rankings.

If query \( q = (K, \tau) \) contains a temporal constraint \( \tau \), this constraint is applied as a filtering condition when traversing the lists. Post times and temporal constraints are translated to numerical values, to improve the efficiency of the filtering action. The temporal values are stored as a numerical field in the inverted index engine, and a Trie prefix tree supports the temporal-condition evaluation.

The relevant cells are extracted from the array \( C_{idx} \), where, for \( 1 \leq j \leq n \), if \( C_{idx}[j][1] \neq \bot \) & \( C_{idx}[j][2] \neq \bot \), the difference \( C_{idx}[j][2] - (C_{idx}[j][1] - 1) \) is the number of relevant posts in Cell \( j \). A linear scan of \( C_{idx} \) is sufficient to compute the relevance measures for each cell, where a Fibonacci heap is used to store the top-\( k \) cells during the scan.

5 VISUALIZATION

The areas of the result are computed based on the discovered cells. First, “noise” should be reduced, i.e., ignoring cells that contain just a few posts. In our implementation, we used a threshold of 5, and removed from the result cells with less than 5 relevant posts. In addition to that, from the top-\( k \) retrieved cells, \( C_1, \ldots, C_k \), we select the lowest \( j \) such that the top-\( j \) cells \( C_1, \ldots, C_j \) satisfy \( \sum_{i=1}^{j} \mu(C_i) \geq 0.8 \sum_{i=1}^{k} \mu(C_i) \), for the ranking measure \( \mu \). Only the top-\( j \) cells are returned. For example, suppose that \( \sum_{i=1}^{k} \mu(C_i) = 1 \) and \( \mu(C_1) = 0.82 \), then only \( C_1 \) is returned, because its score already exceeds 80% of the total sum of scored. If the scores are 0.35, 0.28, 0.22, 0.07, 0.03, 0.01 then the cells with scores 0.35, 0.28, 0.22 are selected because their scores, combined, exceed 80% of the sum of all scores. This reduces “noise” caused by cells with relatively low scores.

Computing Relevant Areas. After finding the relevant cells, we compute the areas—convex polygons with a high relevance score—to provide higher accuracy and finer granularity than cells. This step must rely on the first step of discovering the cells, because it is relatively complex and computationally expensive, as there are many ways to divide the search region into polygonal areas. Furthermore, it is difficult to calculate the scoring functions on a random convex polygonal area because we must know the total number of posts inside the polygonal area.

To compute the polygonal areas, we apply the following two steps. First, adjacent relevant cells from the list \( l \) (the result of the first phase) are merged to create cell groups. This is somewhat similar to a bottom-up hierarchical clustering of cells. Second, for each cell group we apply the OPTICS clustering algorithm [2] to all the posts in that group.

The algorithm receives two parameters—MinPts, which is the minimal number of points in a cluster, and \( \varepsilon \), which is the maximum distance between points to consider when adding a point to a cluster. We set MinPts=4, and \( \varepsilon \) to be the square root of the density of the most dense cell in the group. OPTICS creates a hierarchy of clusters.

For each cluster, we calculate the convex hull of its points using Graham’s scan algorithm [9].

To prevent overlapping clusters, from each branch of the hierarchical clustering tree at most one cluster is selected. We choose the level of the hierarchical clustering tree that maximizes

\[
\text{Number of messages in clusters} = \frac{\text{Number of messages in clusters}}{\text{Area size} \cdot (1 + \log(\text{number of clusters}))}
\]

This balances the conflicting goals of (a) covering as many relevant messages as possible, (2) having a precise cover where the area is as small as possible, and (3) having a small number of clusters.

To prevent noise, we discard areas whose size is below a given threshold. By doing so, we aim to maximize the total number of posts in a cluster while minimizing the overall area and the number of clusters. For optimization purposes, if there are too many posts in a cell, the algorithm is applied to a uniform sample of the posts.

6 EXPERIMENTAL EVALUATION

We present now our experimental evaluation. The goals of the tests are to compare the accuracy of the different ranking methods in detecting significant areas and events, and to illustrate the efficiency of a search using the partition-aware inverted index.

6.1 Setup

We tested our search engine over a dataset of 45 million tweets and Instagram posts that were collected as part of the CityBeat project [27]. This dataset is restricted to the area of New York City.

We selected a grid partition with cell height of 350 meters and cell width of 250 meters, which are 0.003 degree longitude and latitude, in the decimal degree format, in the NYC area. We used a Macbook pro, 2.8 GHz Intel Core i7, 16 GB 1600 MHz DDR3 to show that a search can be executed on a standard machine. The indexing process of the data took about 2 hours, hence, indexing could be done frequently (e.g., every day) to reflect recent changes.

6.2 Accuracy

We tested the accuracy of the different methods, comparing them to one another and to Google Maps. We compiled a list of queries in four different domains: locations, events, activities, and sentiments. The separation to domains enables us to evaluate the accuracy in different types of search.

6.2.1 Event Discovery. To test the accuracy of the methods for event discovery, we collected information about ten public events in NYC, from http://www.timeout.com/. Based on their maps, we annotated the spatial region of each event. For example, for a parade, we annotated the street segments that constitute the parade route. These annotated maps are the ground truth.

We executed geosocial search for each one of the events, and compared the different ranking methods. We also executed the search on Google Maps.

Definition 6.1 (Precision). Given a ground truth as an annotated map with a set \( G \) of polygonal shapes, and a set of polygonal areas \( A \) that are the result of a geosocial search, the precision of the search result is the percentage of polygonal areas that intersect with the annotated events, i.e.,

\[
\frac{|\{A \in A | A \cap G \neq \emptyset\}|}{|A|}
\]
Evaluating the search results of Google Maps required a lenient definition of precision, because Google returns points of interest (POI) rather than regions.

**Definition 6.2 (Lenient Precision).** Given a ground truth as an annotated map with a set $\mathcal{G}$ of polygonal shapes, and a set $P$ of points as a result of a geographic search, the lenient precision of the search result is defined as follows. If at least one of the points in $P$ is relevant and is contained in at least one of the areas of $\mathcal{G}$, then we consider the precision to be 1. If at least one of the POIs of $P$ is irrelevant but is contained in an area of $\mathcal{G}$, then the precision is 0. Otherwise, the precision is 0.

Fig. 7 illustrates a search for “Easter parade” using Global Ratio. The result area (red) is overlaid on the expected Easter parade area (blue). Fig. 8 illustrates a search for “Easter parade” on Google Maps. There were no results, so the precision of the Google search is 0.

We computed the precision score for all the events in Table 1. The first observation is that the precision score of geosocial search in each of the methods is higher than Google’s score. Both Global Ratio and Harmonic Mean, with 85% precision, are better than the other methods in finding events. This aligns with the fact that events tend to engender high activity in the social media, at the location and the time of their occurrence. Local Ratio and Binomial Test perform a little worse, with 80% and 75% precision, respectively. Apparently, statistical significant is not optimal for finding events.

Measuring the volume of posts is more suitable for that cause.

Table 1: Precision scores in event search.

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Local</th>
<th>Global</th>
<th>Harmonic</th>
<th>Binomial Test</th>
<th>Google</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gay Pride Parade</td>
<td>0.66</td>
<td>0.73</td>
<td>0.73</td>
<td>0.69</td>
<td>0.5</td>
</tr>
<tr>
<td>New York Marathon</td>
<td>1</td>
<td>0.9</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Thanksgiving Parade</td>
<td>0.69</td>
<td>0.8</td>
<td>0.81</td>
<td>0.85</td>
<td>0</td>
</tr>
<tr>
<td>Dog Parade</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Reading Room Bryant</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Sakura</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Five Boro</td>
<td>1</td>
<td>0.75</td>
<td>1</td>
<td>0.66</td>
<td>0</td>
</tr>
<tr>
<td>Smorgasburg</td>
<td>0.66</td>
<td>1</td>
<td>1</td>
<td>0.18</td>
<td>1</td>
</tr>
<tr>
<td>Easter Parade</td>
<td>0.5</td>
<td>1</td>
<td>0.5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Mermaid Parade</td>
<td>0.5</td>
<td>0.33</td>
<td>0.5</td>
<td>0.14</td>
<td>1</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>0.8</td>
<td>0.85</td>
<td>0.85</td>
<td>0.75</td>
<td>0.55</td>
</tr>
</tbody>
</table>

6.2.2 Locations, Activities and Sentiments. We tested the different ranking methods, and Google search, in searches of locations, activities and feelings. We executed a query $q$, and to each result assigned a binary score—relevant or irrelevant. The precision is the percentage of retrieved results that are relevant.

Relevance of a result is derived from the intent of the user who poses the query. In a location search, a relevant result is the actual location of the searched entity. For example, when searching for “Madison” in NYC area, “Madison Square Garden”, “Madison Square Park”, and “Madison Ave.” are all considered relevant. In activity search, relevant regions are places where that activity is typically done, e.g. for the query “jogging”, relevant places are areas where people jog, like parks, running routes, gyms, etc. A search based on a sentiment is aimed to find places that are associated with specific feelings, i.e., places in which the searched sentiment is often mentioned. For example, a search for “romantic” is expect to find places that people consider to be romantic, like promenades, gardens, bridges, etc. Table 2 presents the precision scores of the different methods.

Table 2: Precision scores for different types of search.

<table>
<thead>
<tr>
<th>Query</th>
<th>Local</th>
<th>Global</th>
<th>Harmonic</th>
<th>Binomial Test</th>
<th>Google</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empire State Bld.</td>
<td>0.83</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Madison Square</td>
<td>0.66</td>
<td>1</td>
<td>1</td>
<td>0.87</td>
<td>1</td>
</tr>
<tr>
<td>Hospital</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Airport</td>
<td>0.87</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Bryant Park</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Highline</td>
<td>0.71</td>
<td>1</td>
<td>1</td>
<td>0.88</td>
<td>1</td>
</tr>
<tr>
<td>Central Park</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.87</td>
<td>1</td>
</tr>
<tr>
<td>Tribeca</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.8</td>
<td>1</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>0.88</td>
<td>1</td>
<td>1</td>
<td>0.91</td>
<td>1</td>
</tr>
</tbody>
</table>

In activity search, relevant regions are places where that activity is typically done, e.g. for the query “jogging”, relevant places are areas where people jog, like parks, running routes, gyms, etc. A search based on a sentiment is aimed to find places that are associated with specific feelings, i.e., places in which the searched sentiment is often mentioned. For example, a search for “romantic” is expect to find places that people consider to be romantic, like promenades, gardens, bridges, etc. Table 2 presents the precision scores of the different methods.

<table>
<thead>
<tr>
<th>Query</th>
<th>Local</th>
<th>Global</th>
<th>Harmonic</th>
<th>Binomial Test</th>
<th>Google</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relaxing</td>
<td>0.42</td>
<td>0.87</td>
<td>1</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>Singing</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.35</td>
</tr>
<tr>
<td>Walking</td>
<td>0.37</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Jogging</td>
<td>0.85</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Running</td>
<td>0.5</td>
<td>0.75</td>
<td>0.87</td>
<td>0.75</td>
<td>0</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>0.63</td>
<td>0.92</td>
<td>0.97</td>
<td>0.95</td>
<td>0.17</td>
</tr>
</tbody>
</table>

In location search, relevant regions are places where that activity is typically done, e.g. Google Maps has a very low precision in activity search. For example, when searching for “Madison” in NYC area, “Madison Square Garden”, “Madison Square Park”, and “Madison Ave.” are all considered relevant.
NY” or “places to run”, provided similar results. Geosocial search provided more accurate results than Google Maps, with a precision of 0.97 by HM, 0.95 by BT and 0.92 by GR. The results of HM were also more diverse than those of the other methods.

The experiments show that GR results are skewed towards active areas, e.g., Times Square, while HM retrieved regions that also have some local characteristic, yielding more varied results. Fig. 13, which presents the result of a query “walking”, illustrates this. In this figure, the red and pink regions were discovered by GR. The pink, white, and blue regions were returned by HM, and the blue regions were selected by LR. While GR returned mostly results in Time Square, HM provided more diverse results, including Central Park, High Line, Washington Square, Brooklyn Bridge, and Williamsburg Bridge.

In sentiment-based search, like in activity search, Google Maps could not find relevant places. Geosocial search, in comparison, was proven to be successful. For example, when posing the query “admiring”, as depicted in Fig. 14, relevant places were discovered, including Metropolitan Museum of Art, Museum of Modern Art (MoMA), Top of The Rock Observation Deck, Bryant Park, Empire State Building and more (the red and pink regions were discovered by GR; the pink, white, and blue regions were selected by HM; the blue regions were returned by LR). In this domain, as well, the best precision was of the methods GR and HM.

In conclusion, traditional geographic search engines, like Google Maps, are successful at finding well-known entities directly by name. However, when trying to find places by their “social” features, like what people do there, how people feel there or by the name of an event that occurs there, geosocial search is more effective than traditional geographic search. Among the models we tested, HM provides the most accurate results, and in particular, it outperforms ranking based on the Binomial statistical test.

6.3 Efficiency

We discuss now the performances of our methods, and we compare the running times of the algorithms LC, BC, and IBC. To test the efficiency of the algorithms, we assembled a list of 50 queries. The selected queries were chosen to provide diversity in the number of relevant posts—from the query “tai chi”, with merely 198 relevant posts, to the query “my”, with 4,325,394 relevant posts.

For each query, we retrieved the list $L$ of relevant posts, executed each algorithm on $L$, and measured the time (in millisecond) to count the number of posts in each cell. We conducted each experiment 10 times, and we report the average of the results. Fig. 15 depicts the running times of the algorithms.

The computation time of LC grows linearly with the number of posts. The BC algorithm becomes more efficient than LC once the number of posts exceeds 75,000. However, the IBC algorithm has the best performances across all the data. It is 50% more efficient than BC for lists of any length, keeping the computation time below 50 milliseconds, compared to more than 100 milliseconds when using the others algorithms.
Figure 15: Running times (in milliseconds) when counting the number of posts in each cell (avg. of 10 runs).

7 CONCLUSION

This paper presents and investigates geosocial search—a spatio-temporal free-text search that returns relevant areas based on the plethora of geotagged posts in the social media. Geosocial search associates what-people-say on social media with the location and the time of sharing; and it provides useful results that ordinary geographic search tools do not discover.

Geosocial search requires processing numerous posts, finding relevant areas, and ranking discovered areas, without knowing a priori the set of all possible areas. To cope with that, we presented a two-step approach. The search area is partitioned into cells, arbitrarily. First, relevant cells are retrieved. Then, result areas are computed by clustering relevant posts in the discovered cells, and computing the convex hull of the clusters.

We introduced different models for evaluating the relevance of discovered areas, to rank the areas of the result. This includes Global Ratio (GR), Local Ratio (LR) and Harmonic Mean (HM) of GR and LR. We also presented a measure based on Binomial test (BT)—a standard non-parametric statistical test—for comparison.

We show experimentally that for various types of search, like searches for events or activities, Geosocial search is more effective than ordinary geographic search, like that of Google Maps. The experiments show that HM is the most effective method, GR is somewhat less effective, LR and BT are the least effective methods.

We use a partition-aware inverted index and the IBC algorithm, to efficiently count the number of relevant posts in each cell, when evaluating queries. Using the index and the IBC algorithm, queries are computed in a fraction of a second over more than 40 million posts, even on an ordinary machine.

Future work includes testing the effect of different spatial partitions of the search area on the quality of the search result.

REFERENCES


