IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 30, 2018, accepted January 22, 2019, date of publication March 4, 2019, date of current version March 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2896934

An Improved Density-Based Approach to Spatio-Textual Clustering on Social Media

MINH D. NGUYEN[1] AND WON-YONG SHIN[2], (Senior Member, IEEE)

[1]Computer, Electrical and Mathematical Science and Engineering Division, King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia
[2]Department of Computational Science and Engineering, Yonsei University, Seoul 03722, South Korea

Corresponding author: Won-Yong Shin (wy.shin@yonsei.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) funded by the Ministry of Education through the Basic Science Research Program under Grant 2017R1D1A1A09000835.

ABSTRACT Density-based spatial clustering of applications with noise (DBSCAN) is the most commonly used density-based clustering algorithm but may not be sufficient when the input data type is *heterogeneous* in terms of textual description. When we aim to discover clusters of geo-tagged records relevant to a particular point of interest (POI) on social media, examining only one type of input data (e.g., the tweets relevant to a POI) may draw an incomplete picture of clusters due to noisy regions. To overcome this problem, we introduce DBSTexC, a newly defined density-based clustering algorithm using *spatio-textual* information on social media (e.g., Twitter). We first characterize the POI-relevant and POI-irrelevant geo-tagged tweets as the texts that include and do not include a POI name or its semantically coherent variations, respectively. By leveraging the proportion of the POI-relevant and POI-irrelevant tweets, the proposed algorithm demonstrates much higher clustering performance than the DBSCAN case in terms of $\mathcal{F}_1$ score and its variants. While DBSTexC performs exactly as DBSCAN with the textually homogeneous inputs, it far outperforms DBSCAN with the textually heterogeneous inputs. Furthermore, to further improve the clustering quality by fully capturing the geographic distribution of geo-tagged points, we present *fuzzy* DBSTexC (F-DBSTexC), an extension of DBSTexC, which incorporates the notion of fuzzy clustering into the DBSTexC. We then demonstrate the consistent superiority of F-DBSTexC over the original DBSTexC via intensive experiments. The computational complexity of our algorithms is also analytically and numerically shown.

INDEX TERMS Density-based clustering, fuzzy clustering, geo-tagged record, point-of-interest (POI), spatio–textual information.

I. INTRODUCTION

A. BACKGROUND

Clustering is one of the prominent tasks in exploratory data mining, and a common technique for statistical data analysis. Cluster analysis refers to the partitioning of objects into a finite set of categories or clusters so that the objects in one cluster have high similarity but are clearly dissimilar to objects in other clusters [1]. Several different approaches to clustering have broadly been introduced in the literature. For example, algorithms such as K-means [2] and Clustering Large Applications based on Randomized Search (CLARANS) [3] were designed based on a partitioning approach; Gaussian mixture models [4] and COBWEB [5] belong to a model-based approach; Divisive Analysis (DIANA) [6] and Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) [7] were developed based on a hierarchical approach; Statistical Information Grid (STING) [8] and Clustering in Quest (CLIQUE) [9] were designed as a grid-based approach; and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [10] and Ordering Points to Identify the Clustering Structure (OPTICS) [11] are examples of a density-based approach.

Among those approaches, density-based clustering has been extensively studied to discover insights in geographic data [12]. Due to the fact that density-based clustering returns clusters of an arbitrary shape, is robust to noise, and does not require prior knowledge on the number of clusters, it is suitable for diverse nature-inspired applications [13]. For instance, through density-based clustering on geographic

The associate editor coordinating the review of this manuscript and approving it for publication was Mamoun Alazab.

data, researchers are capable of finding clusters of restaurants in a city, clusters along roads and rivers, and so forth. Due to its robust performance and intuitive representation, DBSCAN stands out as the most frequently used density-based clustering algorithm. Variations of DBSCAN were also widely studied in [12], [14], and [15].

Recently, owing to the popularity of online social networks (or equivalently, social media), the volume of spatio–textual data is rising drastically. Hundreds of millions of users on social media tend to share their geo-tagged media content such as photos, videos, musics, and texts. For example, when users visit a point-of-interest (POI), they are likely to check in, upload photos of their visit, or post geo-tagged textual data via social media to describe their individual idea, feeling or preference relevant to the POI. An example includes the case where more than five hundred million tweets are posted on Twitter [16] everyday,[1] and approximately 1% of them are geo-tagged [17], which correspond to five million geo-tagged tweets everyday. As a result, there is a high demand for processing and making good use of spatio–textual information based on massive datasets of real-world social media. While there were several studies on the spatio-textual queries [18]–[21], which are to find objects satisfying certain spatial and textual constraints, researches on spatio-textual data analysis by clustering [22], [23] have not been closely and comprehensively carried out.

B. MOTIVATION AND MAIN CONTRIBUTIONS

Our study is motivated by the insight that when we find clusters (or geographic regions) from geo-tagged records related to a certain POI on social media, DBSCAN [10] and its several variations [12], [14], [15] may not provide good clustering results. This comes from the fact that while the geographic region surrounding a POI generally comprises two types of *heterogeneous* geo-tags that include and do not include annotated keywords about the POI (defined as *POI-relevant* and *POI-irrelevant* geo-tags, respectively), DBSCAN uses only one type of input data (e.g., POI-relevant geo-tags) in the process of finding clusters. Therefore, although clusters found by DBSCAN seem to correctly discover groups of POI-relevant geo-tags on the surface, they also blindly include geographic regions which contain a large number of undesired POI-irrelevant geo-tags, thus leading to a poor clustering quality. Hence, in the case of such a heterogeneous input data type, the methodology of DBSCAN using only POI-relevant geo-tags may not be a complete solution to finding clusters. It is essential to perform clustering based on a textually heterogeneous input, including both POI-relevant and POI-irrelevant geo-tagged records, in order not only to find highly dense clusters of POI-relevant geo-tagged points but also to exclude the regions with a large number of POI-irrelevant points.

To this end, we introduce DBSTexC, a novel spatial clustering algorithm based on *spatio–textual* information on social media such as Twitter [24], [25].[2] We first characterize POI-relevant and POI-irrelevant geo-tagged tweets as the texts that include and do not include a POI name or its semantically coherent variations, respectively. By judiciously considering the proportion of both POI-relevant and POI-irrelevant tweets, DBSTexC is shown to greatly improve the clustering quality in terms of $\mathcal{F}_1$ score and its variants including a geographic factor, compared to that of DBSCAN. This gain comes due to the robust ability of DBSTexC that excludes noisy regions which contain a huge number of undesired POI-irrelevant tweets. Note that DBSTexC can be regarded as an extension of DBSCAN since it performs exactly as DBSCAN with the textually homogeneous inputs and far outperforms DBSCAN with the heterogeneous inputs.

It is worth noting that DBSTexC assumes the resulting clusters having strict boundaries, which however may not fully exploit the entire geographic features of the data. To further improve the clustering quality based on the observation that the geographic distribution of tweets is generally smooth and thus it is not clear which tweets should be grouped as clusters or be treated as noise, we present a *fuzzy* DBSTexC (F-DBSTexC) algorithm. F-DBSTexC relaxes the constraints on a point's neighborhood density by allowing an ambiguous tweet to belong to a cluster with a distinct membership degree. We empirically evaluate its performance by showing the superiority over the original DBSTexC in terms of our performance metric. This additional gain over the original DBSTexC comes from the fact that decision boundaries for clusters can be fuzzy. The runtime complexity of our two algorithms is also analytically shown, and our analysis is numerically validated. Our main contributions are five-fold and summarized as follows:

- We introduce DBSTexC, a new spatial clustering algorithm, which intelligently integrates the existing DBSCAN algorithm and the heterogeneous textual information to avoid geographic regions with a large number of POI-irrelevant geo-tagged posts in the resulting clusters.
- We show the evaluation performance of the proposed clustering algorithm in terms of $\mathcal{F}_1$ score and its variants, while demonstrating its superiority over DBSCAN by up to about 60%.
- We also present the F-DBSTexC algorithm, an extension of DBSTexC, which incorporates the notion of fuzzy clustering into the DBSTexC framework, to fully capture the geographic distribution of tweets in various locations.
- We demonstrate the robust ability of F-DBSTexC that further improves the clustering quality via intensive experiments, compared to that of DBSTexC by up to about 27% for several POIs that are located especially in sparsely-populated areas.

[1] www.internetlivestats.com/ accessed on December 26, 2018.

[2] Even if our focus is on analyzing tweets, the dataset on other social media (or micro-blogs) can also be directly applicable to our research.

TABLE 1. Summary of notations.

Notation	Description
ϵ	Radius of a point's neighborhood
N_{min}	Minimum allowable number of POI-relevant tweets in an ϵ-neighborhood of a point
N_{max}	Maximum allowable number of POI-irrelevant tweets in an ϵ-neighborhood of a point
η	Precision threshold for a query region
$\mathcal{X}$	Set of POI-relevant tweets
$\mathcal{Y}$	Set of POI-irrelevant tweets
$\mathcal{X}_\epsilon(p)$	Set of POI-relevant tweets contained in an ϵ-neighborhood of point p
$\mathcal{Y}_\epsilon(p)$	Set of POI-irrelevant tweets contained in an ϵ-neighborhood of point p
$\text{dist}(p, q)$	Euclidean distance between points p and q
C	A cluster with label C
A	Area of the geographical region covered by clusters
$\bar{A}$	Normalized area of the geographical region covered by clusters
α	Area exponent
$\mathcal{F}_1$	$\mathcal{F}_1$ score
n	Number of POI-relevant tweets
m	Number of POI-irrelevant tweets
μ_p	Fuzzy score of point p

- We analytically and numerically show the computational complexity of our proposed algorithms when two different implementation approaches are employed.

This paper is the first attempt to integrate the existing DBSCAN and the heterogeneous textual information, and thus our methodology sheds light on how to design highly-improved spatial clustering algorithms by leveraging spatio–textual information on social media.

C. ORGANIZATION

The rest of the paper is organized as follows. In Section II, we review the prior work related to our research. Section III describes how to collect POIs and search for POI-relevant tweets. In Section IV, we present the proposed DBSTexC algorithm and empirically evaluate its performance. The computational complexity of our algorithm is analytically shown in Section V. Section VI introduces F-DBSTexC, an extended version of DBSTexC. Finally, Section VII summarizes the paper with some concluding remarks.

D. NOTATIONS

The list of all the notations used in our work is presented in Table 1. Some notations will be more precisely defined as they appear in later sections of this paper.

II. PREVIOUS WORK

Our clustering algorithm is related to four broad areas of research, namely traditional spatial clustering, spatio–textual similarity search, clustering based on spatial and non-spatial attributes, and fuzzy clustering.

A. SPATIAL CLUSTERING

A variety of spatial clustering algorithms have been developed in the literature. Several algorithms using a partitioning approach were introduced and widely utilized in [2], [3], and [26]. Even though such algorithms are useful for finding sphere-shaped clusters, they require prior knowledge on the number of clusters and thus are unable to find clusters of arbitrary shapes. Next, hierarchical clustering algorithms [6], [7] can be further divided into two types based on the following clustering processes: the agglomerative (bottom-up) process and the divisive (top-down) process. Their strengths lie in the hierarchical relation among clusters and an easy interpretation. However, hierarchical clustering does not have well-defined termination criteria, and if some objects are mis-clustered during the growth of the hierarchy, then such objects will remain in a certain wrong cluster until the clustering process is terminated. In addition, from a density-based point of view, the DBSCAN algorithm [10] uses a series of density-connected points to form density-based clusters. Since DBSCAN does not require the number of clusters as an input parameter, and does not assume any underlying probability density behind the clusters, it can discover clusters of arbitrary shapes. As follow-up studies on DBSCAN, numerous algorithms have been developed as follows: GDBSCAN [12] generalized DBSCAN by extending the notion of a neighborhood over the traditional ϵ-neighborhood and by using different measures to define the "cardinality" of the neighborhood; ST-DBSCAN [14] was designed by discovering clusters based on spatial and temporal attributes; HDBSCAN [15] was presented by generating a density-based clustering hierarchy and then extracting a set of significant clusters based on a measure of stability. Unlike the aforementioned studies, our work aims to integrate the existing DBSCAN and the heterogeneous textual information to avoid noisy regions having numerous POI-irrelevant geo-tags.

B. SPATIO–TEXTUAL SIMILARITY SEARCH

It is of paramount importance to find spatially and textually closest objects to query objects. To offer compelling solutions to this problem, several algorithms [18]–[21] were introduced. Particularly, a method to answer queries containing a location and a set of keywords was presented in [18]. Next, an indexing framework for processing top-*k* query that takes into account both spatial proximity and text relevancy was introduced in [19]. Although these algorithms study the spatio–textual distance between objects, they are inherently different from our proposed approach, which finds density-based spatio–textual clusters using the textually heterogeneous input data type on social media such as Twitter.

C. CLUSTERING BASED ON SPATIAL AND NON-SPATIAL ATTRIBUTES

There have been recent studies on the use of spatial and non-spatial attributes to improve the clustering performance in various applications. Spectral clustering was applied in [27] to identify clusters among gang members based on both the observation of social interactions and the geographic locations of individuals. On the other hand, another clustering method was presented in [28] to discover clusters that are dense spatially and have high spatial correlation based on their non-spatial attributes.

D. FUZZY CLUSTERING

Most of fuzzy clustering algorithms were built upon the fuzzy c-means algorithm [29]–[31]. These algorithms integrate crisp clustering techniques and the theory of fuzzy sets so as to discover clusters whose objects belong to multiple clusters simultaneously with different degrees of membership [32], [33]. However, fuzzy density-based clustering algorithms may or may not allow overlapping clusters. Fuzzy neighborhood DBSCAN (FN-DBSCAN) [34] was proposed by introducing the definition of the fuzzy neighborhood size along with various neighborhood membership functions to capture different neighborhood sensitivities. FDBSCAN [35] introduced fuzzy distance functions to express the similarity between two objects and integrated these functions into DBSCAN. Three extensions of DBSCAN were also presented in [36], while producing clusters with distinct fuzzy and overlapping properties. A survey on popular fuzzy density-based clustering algorithms was presented in [37].

III. DATA ACQUISITION AND PROCESSING

We first explain how we acquire the Twitter data and choose POIs. Then, for every POI, we outline our approach to searching for POI-relevant and POI-irrelevant geo-tagged tweets.

A. COLLECTING TWITTER DATA

We utilize the Twitter Streaming Application Programming Interface (API) [38], which is a widely popular tool to collect data from Twitter for various research purposes such as topic modeling, network analysis, and statistical content analysis. Streaming API returns tweets that match a query written by an API user. An interesting finding is that even if Twitter Streaming API returns at most a 1% sample of all the tweets created at a given moment, it gives an almost complete set of *geo-tagged* tweets despite sampling [17].

The dataset that we use includes 946,801 geo-tagged records (i.e., tweets) collected from 132,342 Twitter users from May 31, 2016 to June 30, 2016 in the UK. We deleted the content objects that were generated by the users posting more than three times consecutively at the same exact location, as those were likely to be products of other services such as Tweetbot, TweetDeck, Twimight, and so forth. Moreover, we notice that each record consists of a number of attributes that can be distinguished by their associated field names. For data analysis, we select the following three attributes from the collected tweets:

- *text*: actual UTF-8 text of the tweet;
- *lat*: latitude of the location where the tweet was posted;
- *lon*: longitude of the location where the tweet was posted.

B. COLLECTING POIS

We select POIs as popular point locations which people may be interested in and are likely to visit. Moreover, for the geographic diversity, we choose POIs from both populous metropolitan areas and sparsely populated cities. The names of chosen POIs and their geographic regions are shown in Table 2. Based on the UK gridded population dataset [39], we are able to approximate the population as follows: the population density for the areas surrounding POIs in London, Edinburgh, and Oxford is >7,000/km^2, <2,000/km^2, and <1,000/km^2, respectively.

TABLE 2. POI names and the corresponding geographic regions.

POI name	Region
Hyde Park	Populous metropolitan area
Regent's Park	Populous metropolitan area
University of Oxford	Sparsely populated city
Edinburgh Castle	Sparsely populated city

TABLE 3. POI names and their search queries.

POI name	Search queries
Hyde Park	*Hyde Park, Kensington Gardens, Royal Park*
Regent's Park	*Regent's Park, London Zoo, tasteoflondon*
University of Oxford	*Oxford Univ, oxford univ, Univ Oxford*
Edinburgh Castle	*Edinburgh Castle, edinburgh castle, EdinburghCastle*

C. SEARCHING POI-RELEVANT TWEETS

Since Twitter users tend to convey their interest in a POI by mentioning or tagging it in their tweets, we are able to collect all POI-relevant tweets by querying for keywords related to the POI in the text field of the collected tweets. However, when users type the actual terms of each POI in their tweets, they may misspell or implicitly mention the POI name. We thus implement a keyword-based search for *semantically coherent* variations of a POI, which would contain its shortened names, its informal names (if any), and so forth.[3] For a POI formed into a large geographic area, we include names of famous attractions inside the POI to increase the search accuracy. The list of search queries for four POIs shown in Table 2 is summarized in Table 3. Therefore, the dataset can be divided into two subgroups of geo-tagged tweets that include and do not include the annotated POI keywords, which correspond to POI-relevant and POI-irrelevant geo-tagged tweets, respectively.

IV. PROPOSED METHODOLOGY

To elaborate on the proposed methodology, we first present the important definitions and analysis that are essential to the design of our algorithm, and show the analysis that validates the correctness of our algorithm. Then, we elaborate on our DBSTexC algorithm.

A. DEFINITIONS

We start by introducing the definition of a query region. A query region is defined as a geographic area from which

[3] Note that brown clustering can also be adopted to find semantically coherent variations of a POI, even if it is not taken into account in our study.

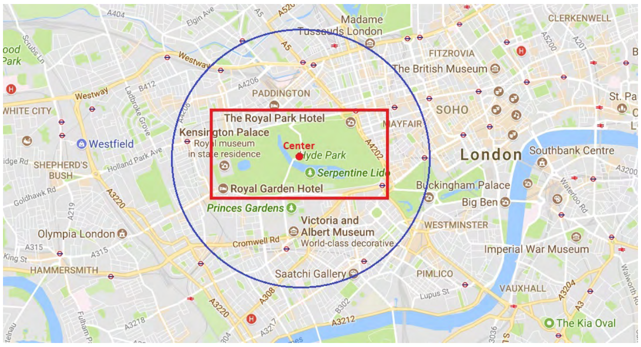


FIGURE 1. An example of the query region for Hyde Park. The red rectangle is the administrative bounding box, whose center is denoted by the red dot, and the blue circle is the query circle that fulfills the condition in Definition 1.

we collect the geo-tagged tweets for a particular POI. We aim at finding both POI-relevant and POI-irrelevant tweets inside the region. Nevertheless, since the relevance of information to the POI varies according to the geographic distance between the POI and the locations where the data are generated, tweets posted at locations far away from the POI are likely to have little or no textual description for the POI. We thus focus only on a region that contains almost all relevant tweets but omit the majority of irrelevant tweets that were posted geographically far from the POI, which would lead to a reduced computational complexity. Motivated by this observation, we define a query region as follows:

Definition 1 (Query Region): Given a POI, a query region is a circle whose center corresponds to the center point of the POI's administrative bounding box provided by Google Maps. The radius of the circle is then increased stepwise until Precision of the query region is lower than a threshold η, where η can be set appropriately based on POI types, which will specified in Section V-B. Here, Precision of the query region is the ratio of true positives (the number of POI-relevant tweets in the query region) to all predicted positives (the number of all retrieved geo-tagged tweets in the query region).

In Fig. 1, we illustrate an example of the query region for Hyde Park. As shown in the figure, starting from the center of the POI, we continue on expanding the query region until the condition in Definition 1 is fulfilled.

Similarly as in DBSCAN [10], we exploit the neighborhood of a point (See Definition 2) and a series of density-connected points (See Definition 6) to find clusters. However, unlike DBSCAN, we present a new parameter $N_{\max}$ to limit the number of *POI-irrelevant* tweets, resulting in an improved clustering quality. Hence, we can acquire a core point which has not only at least $N_{\min}$ POI-relevant tweets but also at most $N_{\max}$ POI-irrelevant tweets inside its neighborhood (See Definition 3). The result of DBSTexC, whose clusters are composed of connected neighborhoods of core points, would be expected to significantly outperform DBSCAN that uses only POI-relevant tweets, which is numerically shown in Section V.

Definition 2 (ϵ-Neighborhood of a Point): Let $\mathcal{X}$ and $\mathcal{Y}$ denote the sets of POI-relevant and POI-irrelevant tweets, respectively. For a point $p \in \mathcal{X}$, the sets of ϵ-neighborhoods containing POI-relevant and POI-irrelevant tweets, denoted by $\mathcal{X}_\epsilon(p)$ and $\mathcal{Y}_\epsilon(p)$, are defined as the geo-tagged tweets within a scan circle centered at p with radius ϵ and are given by

$$\mathcal{X}_\epsilon(p) = \{q \in \mathcal{X} | \text{dist}(p, q) \leq \epsilon\}$$
$$\mathcal{Y}_\epsilon(p) = \{q \in \mathcal{Y} | \text{dist}(p, q) \leq \epsilon\},$$

respectively, where $\text{dist}(p, q)$ is the geographic distance between coordinates p and q. Note that we focus on the ϵ-neighborhood only for POI-relevant tweets while neglecting the neighborhood of POI-irrelevant tweets, since our DBSTexC algorithm finds clusters based on a series of ϵ-neighborhoods of only POI-relevant tweets.

Definition 3 (Core Point): A point $p \in \mathcal{X}$ is a core point if it fulfills the following condition:

$$|\mathcal{X}_\epsilon(p)| \geq N_{\min} \text{ and } |\mathcal{Y}_\epsilon(p)| \leq N_{\max}.$$

B. ANALYSIS

The analytical part essentially follows the same line as that in [12], but is modified so that it fits into our clustering framework. In this subsection, we present fundamental definitions that provide the basis for our DBSTexC algorithm to find clusters according to a density-based approach using spatio–textual information. Then, we analytically validate the correctness of our algorithm by introducing two lemmas.

Definition 4 (Directly Density-Reachable): A point p is directly density-reachable from a core point q with respect to (w.r.t.) ϵ, $N_{\min}$, and $N_{\max}$ if

$$p \in \mathcal{X}_\epsilon(q) \quad \text{or } p \in \mathcal{Y}_\epsilon(q).$$

If point p is directly density-reachable from a point q and is a core point itself, then q is also directly density-reachable from p. Therefore, it is obvious that "directly density-reachable" is symmetric for pairs of core points.

Definition 5 (Density-Reachable): A point p is density-reachable from a point q w.r.t. ϵ, $N_{\min}$, and $N_{\max}$ if there is a chain of points $p_1, \cdots, p_n$, $p_1 = q$, and $p_n = p$ such that p_{i+1} is directly density-reachable from p_i.

The density-reachable relation is not symmetric. For example, given a directly density-reachable chain as in Definition 5, the points $p_1, \cdots, p_{n-1}$ are all core points. However, p_n can be either a border point or a core point. If p_n is a core point, then point p_1 is also symmetrically density-reachable from p_n. Therefore, if the two points p and q are density-reachable from each other, then they are core points and belong to the same cluster.

Definition 6 (Density-Connected): A point p is density-connected to a point q w.r.t. ϵ, $N_{\min}$, and $N_{\max}$ if there is a point o such that both p and q are density-reachable from o w.r.t. ϵ, $N_{\min}$, and $N_{\max}$.

With the above six definitions, we are now ready to define a new notion of a cluster. In brief, a cluster (See Definition 7) is defined as a set of density-connected points. Noise points (See Definition 8) are defined as the set of points not belonging to any clusters.

Definition 7 (Cluster): Let $\mathcal{T}$ denote the dataset of all retrieved geo-tagged tweets. Then, a cluster C w.r.t. ϵ, $N_{\min}$, and $N_{\max}$ is a non-empty subset of the dataset $\mathcal{T}$ satisfying the following conditions:

1) $\forall p \in \mathcal{X}$ and $q \in \mathcal{T}$: if $p \in C$ and q is density-reachable from p w.r.t. ϵ, $N_{\min}$, and $N_{\max}$, then $q \in C$.
2) $\forall p, q \in C$: p is density-connected to q w.r.t. ϵ, $N_{\min}$, and $N_{\max}$.

Definition 8 (Noise): Let $C_1, \cdots, C_k$ be the clusters in the dataset $\mathcal{T}$. Then, noise is defined as the set of points in $\mathcal{T}$ not belonging to any cluster C_i, i.e., $\{p \in \mathcal{T} | p \notin C_i, \forall i\}$.

Given the above eight definitions, our DBSTexC algorithm can then be intuitively stated as a two-step clustering algorithm using spatio–textual information. The first step is to choose an arbitrary POI-relevant tweet satisfying the core point condition as a seed. The second step is to retrieve all points that are density-reachable from the seed, thus obtaining the corresponding cluster containing the seed. To formally justify the credibility of our algorithm, we establish the following two lemmas.

Lemma 1: Let p be a point in $\mathcal{X}$, $|\mathcal{X}_\epsilon(p)| \geq N_{min}$, and $|\mathcal{Y}_\epsilon(p)| \leq N_{max}$. Then, the set $O = \{o | o \in \mathcal{T}$ and o is density-reachable from p w.r.t. ϵ, N_{min}, and N_{max} $\}$ is a cluster w.r.t. ϵ, N_{min}, and N_{max}.

Proof: Since $p \in \mathcal{X}$, $|\mathcal{X}_\epsilon(p)| \geq N_{\min}$ and $|\mathcal{Y}_\epsilon(p)| \leq N_{\max}$, p is a core point and thus is contained in some cluster C. We need to show that $O \subseteq C$. Definition 7-1 indicates that all points that belong to O should also belong to C, resulting in $O \subseteq C$. This completes the proof of this lemma. □

Lemma 2: Let C be a cluster w.r.t. ϵ, N_{min}, and N_{max}. Let p be any point in $C \cap \mathcal{X}$ with $|\mathcal{X}_\epsilon(p)| \geq N_{min}$ and $|\mathcal{Y}_\epsilon(p)| \leq N_{max}$. Then, C is equal to the set $O = \{o | o$ is density-reachable from p w.r.t. ϵ, N_{min}, and $N_{max}\}$.

Proof: We need to show that $C = O$. Similarly as in the proof for Lemma 1, we have

$$O \subseteq C. \tag{1}$$

Therefore, to show that $C = O$, we need to prove that $C \subseteq O$. Let q be an arbitrary point in C. Since $p \in C$, q is density-connected to p from Definition 7-2. It implies that there is a core point $m \in C$ such that p and q are density-reachable from m (see Definition 6). However, p and m are both core points, which represents that p is density-reachable from m if and only if m is density-reachable from p. This shows that q is density-reachable from p, which indicates that $q \in O$. Therefore, it follows that

$$C \subseteq O. \tag{2}$$

From (1) and (2), we finally have

$$C = O,$$

which completes the proof of this lemma. □

Algorithm 1 **DBSTexC**$(\mathcal{X}, \mathcal{Y}, \epsilon, N_{\min}, N_{\max})$

```
Input: $\mathcal{X}, \mathcal{Y}, \epsilon, N_{\min}, N_{\max}$
Output: Clusters with different labels $C$
Initialization: $C \leftarrow 0$; $n \leftarrow |\mathcal{X}|$; $m \leftarrow |\mathcal{Y}|$; $p_i$ is a point in
    the set $\mathcal{X}$
1: for each $p_i$ do
2:     if $p_i$ is not visited then
3:         Mark $p_i$ as visited
4:         $[\mathcal{X}_\epsilon(p_i), \mathcal{Y}_\epsilon(p_i)] = \text{RangeQuery}(p_i)$
5:         if $|\mathcal{X}_\epsilon(p_i)| \geq N_{\min}$ & $|\mathcal{Y}_\epsilon(p_i)| \leq N_{\max}$ then
6:             $C \leftarrow C + 1$
7:             ExpandCluster($p_i$, $\mathcal{X}_\epsilon(p_i)$, $\mathcal{Y}_\epsilon(p_i)$)
```

Algorithm 2 ExpandCluster$(p_i, \mathcal{X}_\epsilon(p_i), \mathcal{Y}_\epsilon(p_i))$

```
Input: $p_i, \mathcal{X}_\epsilon(p_i), \mathcal{Y}_\epsilon(p_i)$
Output: Cluster $C$ with all of its members
1: Add $p_i$ to the current cluster
2: for each point $p_j$ in the set $\mathcal{X}_\epsilon(p_i)$ do
3:     if $p_j$ is not visited then
4:         Mark $p_j$ as visited
5:         $[\mathcal{X}_\epsilon(p_j), \mathcal{Y}_\epsilon(p_j)] = \text{RangeQuery}(p_j)$
6:         if $|\mathcal{X}_\epsilon(p_j)| \geq N_{\min}$ & $|\mathcal{Y}_\epsilon(p_j)| \leq N_{\max}$ then
7:             $\mathcal{X}_\epsilon(p_i) = \mathcal{X}_\epsilon(p_i) \cup \mathcal{X}_\epsilon(p_j)$
8:             $\mathcal{Y}_\epsilon(p_i) = \mathcal{Y}_\epsilon(p_i) \cup \mathcal{Y}_\epsilon(p_j)$
9:     if $p_j$ does not have a label then
10:        Add $p_j$ to the current cluster
11: if $|\mathcal{Y}_\epsilon(p_i)| \neq 0$ then
12:     for each point $q_j$ in the set $\mathcal{Y}_\epsilon(p_i)$ do
13:         if $q_j$ is not visited then
14:             Mark $q_j$ as visited
15:             if $q_j$ does not have a label then
16:                 Add $q_j$ to the current cluster
```

C. DBSTEXC Algorithm

In this subsection, we describe our DBSTexC algorithm that makes use of both POI-relevant and POI-irrelevant tweets. In the clustering process, DBSTexC starts with a random point p_i in $\mathcal{X}$ (i.e., the set of POI-relevant tweets) for $i \in \{1, ..., |\mathcal{X}|\}$ and retrieves all points that are density-reachable from p_i with respect to ϵ, $N_{\min}$, and $N_{\max}$ (See Algorithm 1). If p_i is a core point, then a cluster is formed and expanded until all points that belong to the cluster are included (See Algorithm 2). Otherwise, DBSTexC moves on to the next point in the set of POI-relevant tweets.

In Algorithm 1, RangeQuery() in line 4 is a function that returns points in an ϵ-neighborhood, where it can be implemented using spatial access methods, i.e., *R-trees* and k-d trees. By searching for both POI-relevant and POI-irrelevant points along with two parameters $N_{\min}$ and $N_{\max}$ to determine whether to create a new cluster and/or expand the current cluster (see line 5 of Algorithm 1), our proposed algorithm effectively excludes noisy areas from its clusters.

In Algorithm 2, for every point $p_j \in \mathcal{X}_\epsilon(p_i)$, we explore the ϵ-neighborhood of p_j. If p_j is a core point, then p_j is added to the current cluster and the algorithm continues by appending its neighbors to the neighbor sets $\mathcal{X}_\epsilon(p_i)$ and $\mathcal{Y}_\epsilon(p_i)$. We repeat this process until all the points in the set $\mathcal{X}_\epsilon(p_i)$ are examined. Eventually, when the process is terminated, the points in the set $\mathcal{Y}_\epsilon(p_i)$ are included in our current cluster.

V. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, to evaluate performance of the proposed DBSTexC algorithm in Section IV-C, we present our performance metric, illustrate experimental results, and analyze the overall average computational complexity.

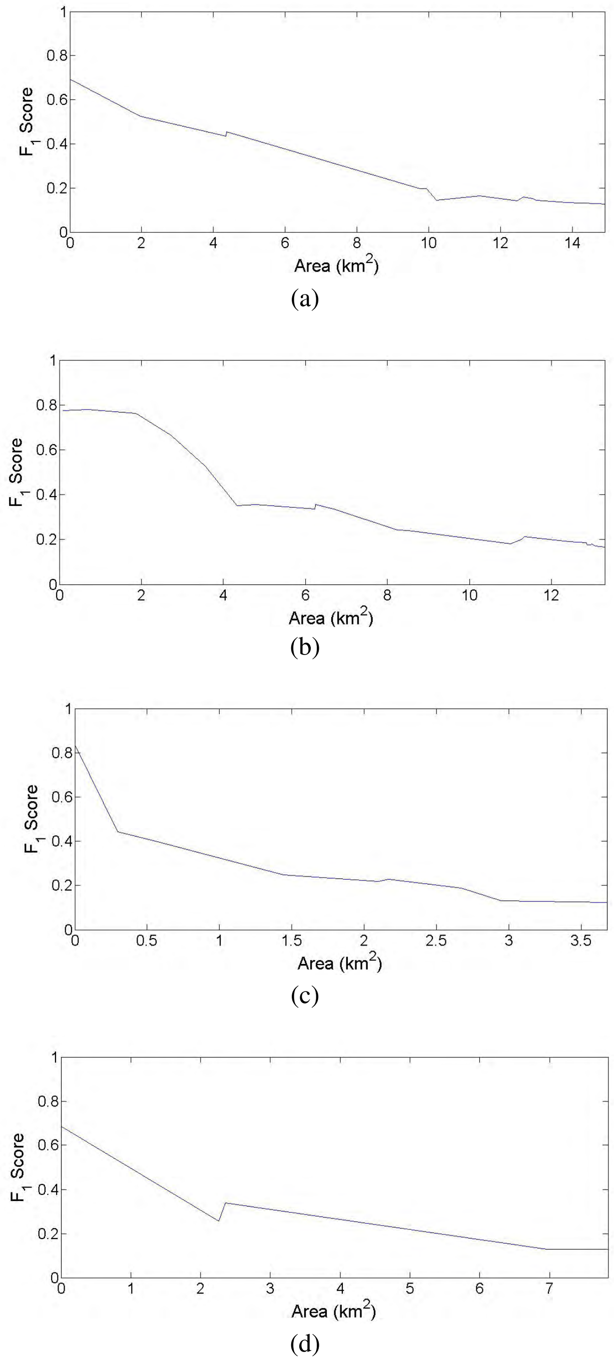
A. PERFORMANCE METRIC

We choose the $\mathcal{F}_1$ score as a key component of our performance metric, since it is a popular measure in machine learning and statistical analysis for a test's accuracy and thus can be a useful tool to assess the clustering quality. The $\mathcal{F}_1$ score is expressed as $\mathcal{F}_1 = 2 \cdot \frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$, which is the harmonic mean of Precision and Recall. In our work, Precision is the ratio of true positives (the number of POI-relevant tweets in clusters) to all predicted positives (the number of all geo-tagged tweets in clusters), that is, $\frac{\text{True Positives (TP)}}{\text{TP+False Positives (FP)}}$; and Recall is the ratio of true positives to actual positives that should have been returned (the total number of POI-relevant tweets), that is, $\frac{\text{TP}}{\text{TP+False Negatives (FN)}}$.

In the process of discovering clusters from geo-tagged tweets relevant to a POI, the area covered by the clusters can be a matter of great interest, since several applications such as geo-marketing may desire a widespread geographic area. To illustrate this point, in Fig. 2, we plot the $\mathcal{F}_1$ score according to the area of the resulting clusters (in km^2) for four chosen POIs. One can observe that the highest $\mathcal{F}_1$ score tends to be found when the area of the resulting clusters is very small. Therefore, although it is good to find clusters with the highest $\mathcal{F}_1$ score, it is more preferred to considerably extend the area of the resulting clusters at the expense of a slightly reduced value of $\mathcal{F}_1$ in some applications. To this end, we would like to formulate a following new performance metric expressed as the product of a power law in the area of the resulting clusters A (in km^2) normalized to the area of the query region, denoted by $\bar{A} = \frac{\text{Area covered by the clusters}}{\text{Area of the query region}}$, and the $\mathcal{F}_1$ score:

$$\bar{A}^\alpha \mathcal{F}_1, \qquad (3)$$

where $\alpha \geq 0$ is the area exponent, which balances between different levels of geographic coverage. When α is small, clusters with almost the highest $\mathcal{F}_1$ score are returned. As a special case, when $\alpha = 0$, our performance metric becomes the $\mathcal{F}_1$ score. On the other hand, as α increases, clusters covering a wide area are obtained at the cost of a reduced $\mathcal{F}_1$. Hence, given parameters for the two algorithms (i.e., $(\epsilon, N_{\min})$ for DBSCAN and $(\epsilon, N_{\min}, N_{\max})$ for DBSTexC), we are able to calculate the performance metric in (3) along with the corresponding $\mathcal{F}_1$ score and the normalized area $\bar{A}$ in each case.



(a)

(b)

(c)

(d)

FIGURE 2. The $\mathcal{F}_1$ score according to the area of the resulting clusters. (a) Hyde Park. (b) Regent's Park. (c) Edinburgh Castle. (d) University of Oxford.

B. EXPERIMENTAL EVALUATION

We exhibit the experimental results for various values of $\alpha \geq 0$. In regard to the query region, for all chosen POIs, we assume that $\eta = 0.07$, which can also be set to other values to control the clustering quality constraint. For the

TABLE 4. Experimental results for DBSCAN and DBSTexC .

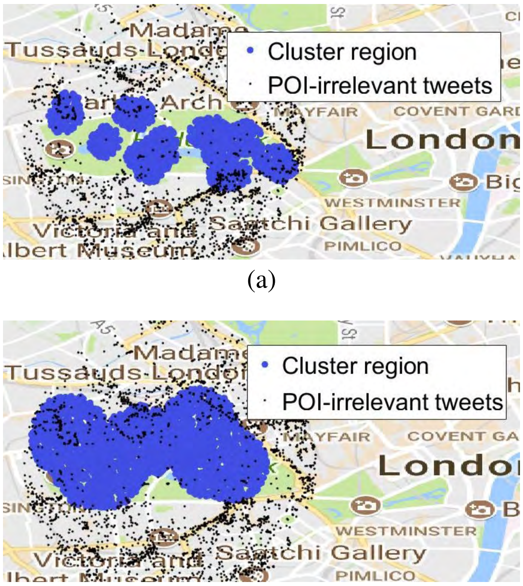
	$A^{\alpha}\mathcal{F}_1$ $(\alpha = 0)$		
POI name	DBSCAN (X)	DBSTexC (Y)	Improvement Rate $\left(\frac{Y-X}{X} \times 100\%\right)$
Hyde Park	0.7333	0.7391	0.79
Regent's Park	0.7795	0.7851	0.72
University of Oxford	0.6930	0.6930	0
Edinburgh Castle	0.8364	0.8364	0
	$A^{\alpha}\mathcal{F}_1$ $(\alpha = 0.5)$		
Hyde Park	0.2103	0.3058	45.41
Regent's Park	0.3184	0.3188	0.13
University of Oxford	0.1288	0.2062	60.09
Edinburgh Castle	0.1333	0.1741	30.61
	$A^{\alpha}\mathcal{F}_1$ $(\alpha = 0.75)$		
Hyde Park	0.1429	0.2284	59.83
Regent's Park	0.2216	0.2219	0.14
University of Oxford	0.1288	0.1673	29.89
Edinburgh Castle	0.1231	0.1510	22.66
	$A^{\alpha}\mathcal{F}_1$ $(\alpha = 1)$		
Hyde Park	0.1253	0.1816	44.93
Regent's Park	0.1303	0.1844	41.52
University of Oxford	0.1288	0.1288	0
Edinburgh Castle	0.1231	0.1412	14.70

parameter set $(\epsilon, N_{\min}, N_{\max})$, since there is no well-known method to determine the best combination with respect to our performance metric in (3), we stepwise test the parameter combinations via exhaustive search. We summarize and compare the performance of both DBSTexC and DBSCAN for four POIs in Table 4, where $\alpha \in \{0, 0.5, 0.75, 1\}$. From the table, it is evident that DBSTexC consistently outperforms DBSCAN in terms of our performance metric in (3) by up to 60.09% for all four chosen POIs. The performance improvement is manifest especially for Hyde Park, which is one of the biggest and the most visited parks in London. In Figs. 3–6, we illustrate the clustering results of DBSCAN and DBSTexC for the four POIs when $\alpha = 0.5$. To emphasize the performance gap between the two algorithms, we depict the geographic cluster region with the distribution of POI-irrelevant tweets. From Fig. 3, one can see that in the Hyde Park case, DBSTexC dramatically excludes a huge number of POI-irrelevant tweets from its clusters, while covering a much bigger geographic area in comparison with DBSCAN. This highlights the robustness of DBSTexC to discover high-quality clusters in terms of the proposed performance metric $\bar{A}^{\alpha}\mathcal{F}_1$.

On the other hand, for a special case where $\alpha = 0$, we notice from Table 4 that the DBSTexC algorithm has almost the same performance as that of DBSCAN. While both algorithms are able to find clusters with the high $\mathcal{F}_1$ score, it is revealed from Fig. 7 that the clusters cover remarkably small geographic areas, which do not provide any insight or useful information about the regions where people are interested in the POIs. As a result, to obtain high-quality clusters covering large geographic areas, it is needed to incorporate the area of clusters into the performance metric.

C. COMPUTATIONAL COMPLEXITY

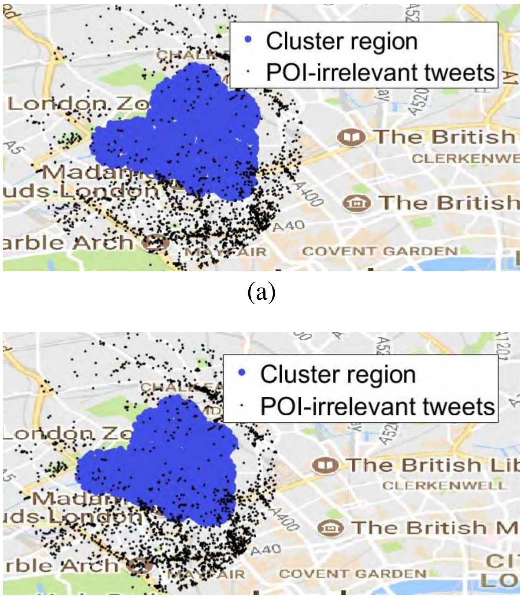
We hereby analyze the computational complexity of the DBSCAN and DBSTexC algorithms. The runtime complexity of both algorithms is calculated as the input size (the number of tweets) times the basic operation ϵ-neighborhood query (range query), which indeed dominates the complexity.



(a)

(b)

FIGURE 3. The results of DBSCAN and DBSTexC for Hyde Park when $\alpha = 0.5$. (a) DBSCAN. (b) DBSTexC.

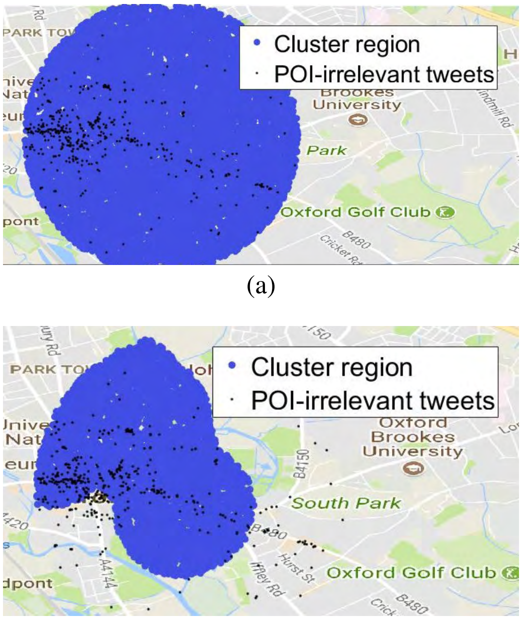


(a)

(b)

FIGURE 4. The results of DBSCAN and DBSTexC for Regent's Park when $\alpha = 0.5$. (a) DBSCAN. (b) DBSTexC.

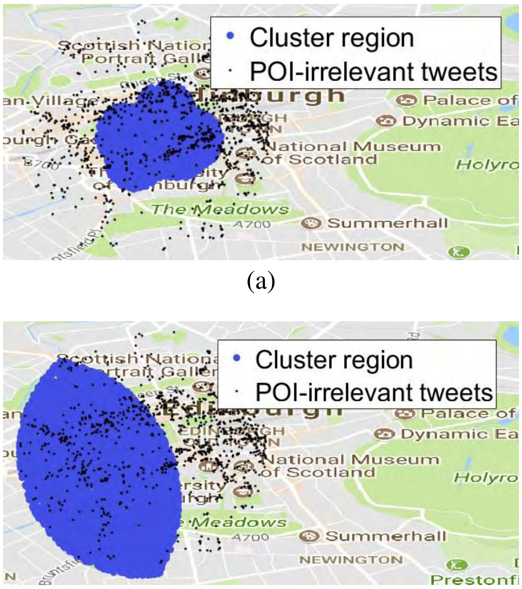
In the case of DBSTexC, from Algorithms 1 and 2, we can clearly see that the RangeQuery() function is invoked only for POI-relevant tweets that have not yet been visited, and the DBSTexC algorithm will visit every POI-relevant tweet in the dataset once. Therefore, we execute exactly one range query for every POI-relevant tweet in the dataset. For analysis, let Q denote the complexity of the function range query, and n and m denote the number of POI-relevant and irrelevant tweets,



(a)

(b)

FIGURE 5. The results of DBSCAN and DBSTexC for University of Oxford when $\alpha = 0.5$. (a) DBSCAN. (b) DBSTexC.



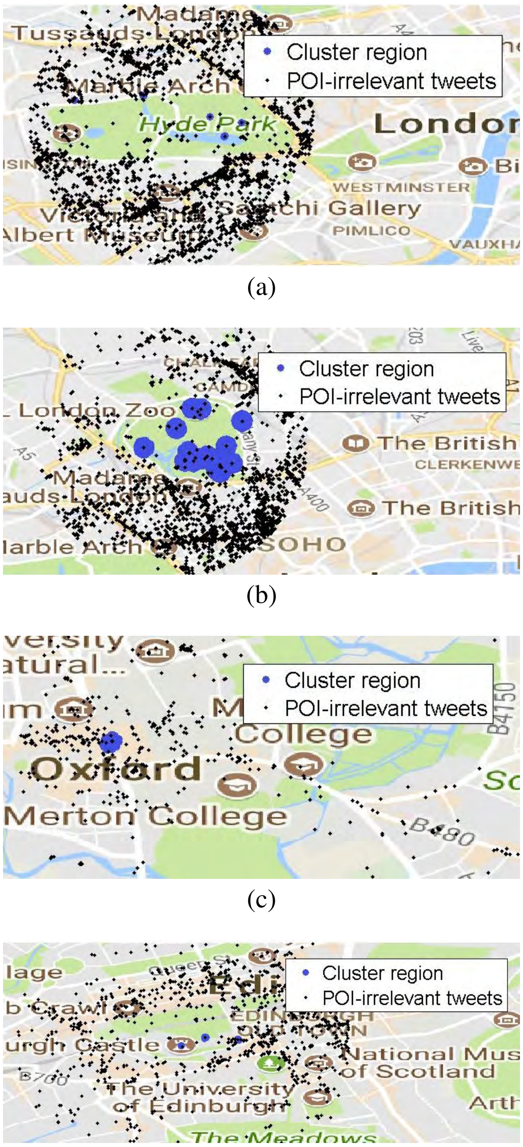
(a)

(b)

FIGURE 6. The results of DBSCAN and DBSTexC for Edinburgh Castle when $\alpha = 0.5$. (a) DBSCAN. (b) DBSTexC.

respectively. It then follows that the complexity is expressed as $\mathcal{O}(n \cdot Q)$. Based on how the function RangeQuery() is implemented, its complexity analysis can be divided into the following two cases:

- If the range query is implemented using a *linear scan*, then we have $Q = \mathcal{O}((n+m) \cdot D)$, where D indicates the cost of computing the distance between two points. Because each geo-tagged tweet in our dataset has a two-dimensional coordinate and is represented by a 64-bit data type in the database, the cost D can be treated as a constant, independent of n and m. Hence, the complexity of the range query and DBSTexC are $\mathcal{O}(n+m)$ and $\mathcal{O}(n^2+nm)$, respectively.



(a)

(b)

(c)

(d)

FIGURE 7. The results of DBSTexC when $\alpha = 0$. (a) Hyde Park. (b) Regent's Park. (c) University of Oxford. (d) Edinburgh Castle.

- If the range query is implemented using a *spatial index*, then we can calculate the worst-case runtime complexity by analyzing both the cost of building the index and the worst-case complexity of the function RangeQuery() used along with the spatial index. For example, for a two-dimensional tree, the worst-case complexity of RangeQuery() is $\mathcal{O}(n+m)$, and the cost of building a two-dimensional tree from $n+m$ geo-tagged points is

$$\begin{aligned}&\mathcal{O}((n+m)\cdot\log(n+m))\\&\quad=\mathcal{O}\left((n+m)\cdot\left(\log n+\log\left(1+\frac{m}{n}\right)\right)\right)\\&\quad=\mathcal{O}((n+m)\cdot\log n),\end{aligned}$$

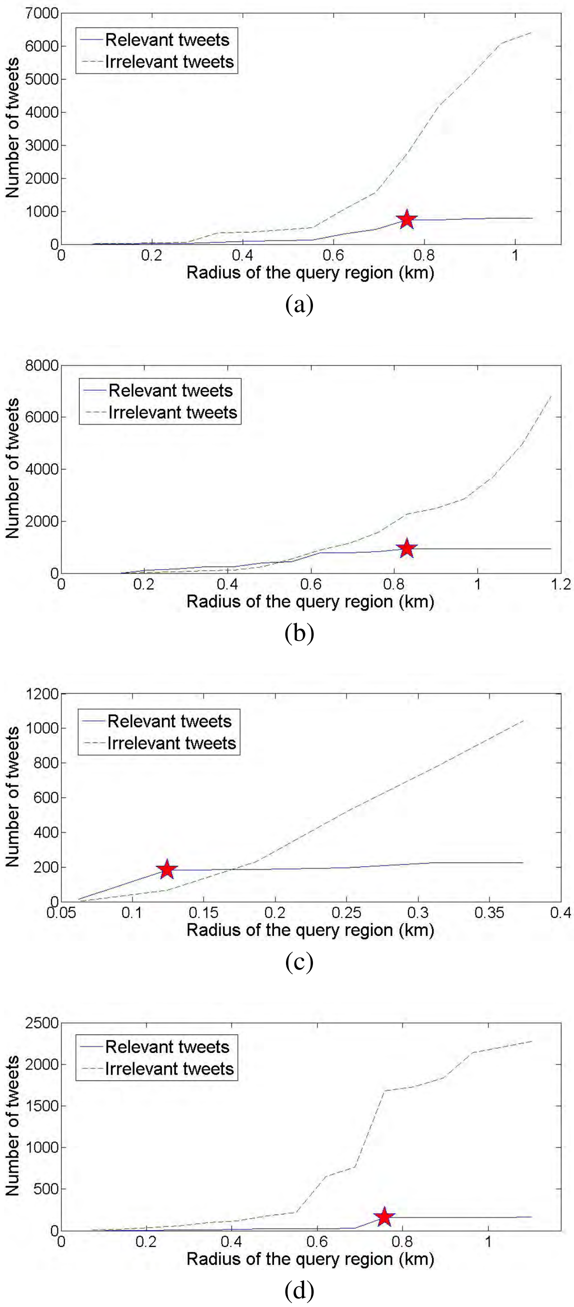
where the last equality holds under the assumption that $m = n^{\beta}$ for $\beta \geq 1$. Therefore, it follows that the time complexity of DBSTexC is $\mathcal{O}(n \cdot (n+m) + (n+m) \cdot \log n) = \mathcal{O}(n^2 + nm)$.

For the DBSCAN algorithm, it has recently been proved in [40] that the worst-case complexity is $\mathcal{O}(n \cdot Q)$. Based on the arguments above, when the range query is implemented using a linear scan, the complexity becomes $\mathcal{O}(n^2 \cdot D) = \mathcal{O}(n^2)$. On the contrary, if the range query is accelerated using a spatial index such as a two-dimensional tree, the worst-case runtime complexity of DBSCAN is $\mathcal{O}(n^2)$ since it takes $O(n \log n)$ to build the tree from n geo-tagged points and the range query has the worst-case complexity of $O(n)$.

To summarize the aforementioned analysis, the worst-case time complexity of DBSTexC and DBSCAN is $\mathcal{O}(n^2 + nm)$ and $\mathcal{O}(n^2)$, respectively. If we focus on a region where $m = c \cdot n$ for a constant $c > 0$, then the complexity of DBSTexC is $\mathcal{O}(n^2)$. In the other region where $m = n^{\beta}$ for $\beta > 1$, the complexity of DBSTexC is $\mathcal{O}(n^{1+\beta})$.

To numerically validate our complexity analysis, we first plot the number of tweets according to different radii of the query region. From Fig. 8, we observe a common trend that the numbers of POI-relevant and POI-irrelevant tweets, denoted by n and m, respectively, increase with the increasing radius of the query region. However, their rates of growth are different; up to a certain radius of the query region, the numbers of POI-relevant and the POI-irrelevant tweets grow at a similar rate, but beyond such a radius (depicted in the figure with a red star), the number of POI-irrelevant tweets grows faster than the number of POI-relevant tweets. This observation is basically consistent with our prior assumption: there is a region where the number of POI-irrelevant tweets is a constant times the number of POI-relevant tweets, having the complexity of $\mathcal{O}(n^2)$ for DBSTexC; and there is another region where the rate of growth of the number of POI-irrelevant tweets is higher than that of the POI-relevant tweets, having the complexity of $\mathcal{O}(n^{1+\beta})$ for $\beta > 1$ for DBSTexC.

We further validate our complexity analysis by plotting the actual runtime complexity of the DBSTexC and DBSCAN algorithms for the worst case. It is easily seen that the worst case takes place when the parameters of DBSTexC and DBSCAN are set to extreme values corresponding to $(\epsilon, N_{\min}) =$ (radius of the query region, 1) for DBSCAN and $(\epsilon, N_{\min}, N_{\max}) =$ (radius of the query region, 1, total number of POI-irrelevant tweets) for DBSTexC. Under this parameter setting, Fig. 9 numerically shows the runtime complexity of the DBSTexC and DBSCAN algorithms in log-log scale according to four different POIs. From Fig. 9, we clearly see that up to a certain value of the number of geo-tagged tweets, DBSTexC and DBSCAN have a similar rate of growth maintaining a constant gap between these two. Beyond the point (depicted in the figure with a red star), the runtime complexity of DBSTexC is higher than that of DBSCAN. Compared with Fig. 8, these transition points exactly match the ones dividing our query region into two sub-regions corresponding to $m = c \cdot n$ for a constant c and $m = n^{\beta}$ for $\beta > 1$. Therefore, from Figs. 8 and 9, it is possible to adequately substantiate our analysis on the complexity of the DBSTexC and DBSCAN algorithms.



(a)

(b)

(c)

(d)

FIGURE 8. **The number of tweets according to the radius of the query region.(a) Hyde Park. (b) Regent's Park. (c) Edinburgh Castle. (d) University of Oxford.**

VI. EXTENSION TO F-DBSTEXC

Thus far, the DBSTexC algorithm has been designed by finding clusters with strict boundaries. For further analysis, we study the geographic distribution of tweets (i.e., two-dimensional coordinates) by using the sorted

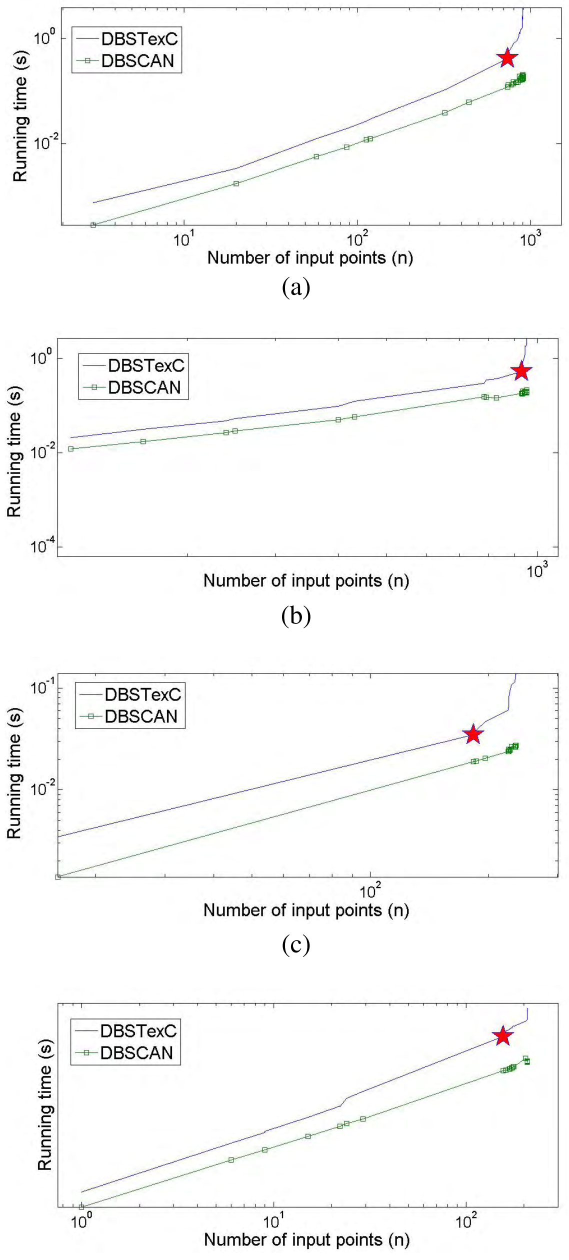


FIGURE 9. The runtime complexity of DBSTexC and DBSCAN. (a) Hyde Park. (b) Regent's Park. (c) Edinburgh Castle. (d) University of Oxford.

k-nearest neighbor (k-NN) distance plot, which shows the distance from geo-tagged points to their k-nearest neighbors sorted in ascending order. If there exists a sudden and sharp increase in the distances between geo-tagged points, then it indicates that clusters and noise points are clearly separated. On the other hand, if we observe a smooth increase in the distances between tweets, then it may not be clear which tweets should be grouped as clusters and which tweets should be treated as noise. In other words, decision boundaries for clusters would be fuzzy. In Fig. 10, the k-NN distance for the four POIs is plotted when $k = 4$. From the figure, we observe that the geographic distribution of tweets is generally smooth. For this reason, using crisp boundaries to separate clusters may not exploit the entire geographic features of the data.

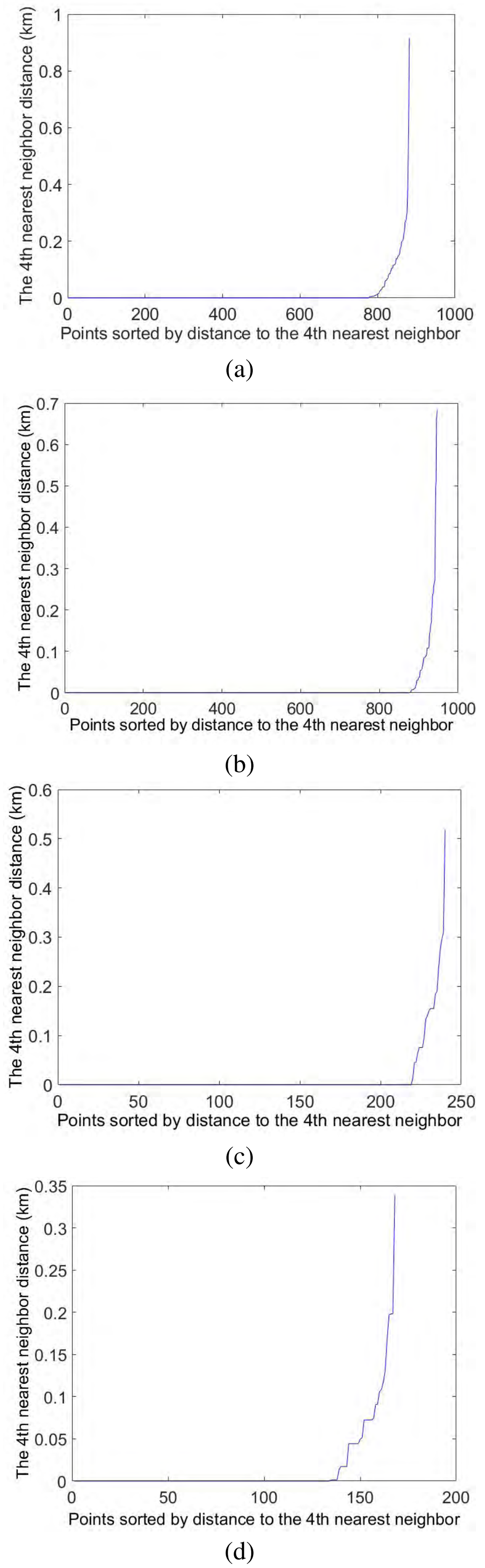


FIGURE 10. The k-NN distance (in km) for different POIs when $k = 4$. (a) Hyde Park. (b) Regent's Park. (c) Edinburgh Castle. (d) University of Oxford.

To overcome this problem, we hereby propose an extension of DBSTexC, called Fuzzy DBSTexC (F-DBSTexC), which incorporates the notion of fuzzy clustering into DBSTexC with a view to fully capturing the geographic characteristics of tweets that tend to be smoothly distributed in space.

A. F-DBSTEXC Algorithm

To design a new algorithm with the notion of fuzzy clustering, we relax the constraints on a point's neighborhood density. That is, we replace the parameters $N_{\min}$ and $N_{\max}$ by two new sets of parameters $(N_{\min_1}, N_{\min_2})$ and $(N_{\max_1}, N_{\max_2})$, respectively, which specify the soft constraints on a point's neighborhood density. For example, in an ϵ-neighborhood of a POI-relevant tweet, if the number of POI-relevant tweets is larger than $N_{\min_1}$ and the number of POI-irrelevant tweets is smaller than $N_{\max_2}$, then a fuzzy neighborhood is generated. To determine the neighborhood cardinality, we introduce monotonically non-decreasing membership functions $J_{Re}(p)$ and $J_{Irre}(p)$ for the POI-relevant tweets and POI-irrelevant tweets, respectively, as follows [36]:[4]

$$J_{Re}(p) = \begin{cases} 1 & \text{if } |X_\epsilon(p)| \geq N_{\min_2} \\ \dfrac{|X_\epsilon(p)| - N_{\min_1}}{N_{\min_2} - N_{\min_1}} & \text{if } N_{\min_1} \leq |X_\epsilon(p)| \leq N_{\min_2} \\ 0 & \text{if } |X_\epsilon(p)| \leq N_{\min_1}, \end{cases} \tag{4}$$

$$J_{Irre}(p) = \begin{cases} 1 & \text{if } |Y_\epsilon(p)| \leq N_{\max_1} \\ \dfrac{N_{\max_2} - |Y_\epsilon(p)|}{N_{\max_2} - N_{\max_1}} & \text{if } N_{\max_1} \leq |Y_\epsilon(p)| \leq N_{\max_2} \\ 0 & \text{if } |Y_\epsilon(p)| \geq N_{\max_2}, \end{cases} \tag{5}$$

where $|X_\epsilon(p)|$ and $|Y_\epsilon(p)|$ denote the number of POI-relevant and POI-irrelevant tweets, respectively, in a neighborhood of point p. These membership functions in (4) and (5) quantify the level of "fuzziness" of a point with respect to clustering. The higher the value of those functions is, the more certain that a point belongs to a cluster. The final cardinality of the ϵ-neighborhood of a point p, μ_p, is then given by[5]

$$\mu_p = \frac{1}{2}[J_{Re}(p) + J_{Irre}(p)]. \tag{6}$$

Based on this notation, the definition of a core point in Definition 3 is revised as below.

Definition 9 (Core Point): A point $p \in \mathcal{X}$ is a core point if it fulfills the following condition:

$$|\mathcal{X}_\epsilon(p)| \geq N_{\min_1} \quad \text{and } |\mathcal{Y}_\epsilon(p)| \leq N_{\max_2}.$$

Next, the F-DBSTexC algorithm is specified in Algorithms 3 and 4. Compared to the original DBSTexC, modified parts correspond to line 5 of Algorithm 3 and line 6 of Algorithm 4, which serve to relax the constraints on a point's neighborhood density. The F-DBSTexC algorithm adds points to the clusters with their distinct fuzzy score μ_p, as expressed in line 9 of Algorithm 4.

Algorithm 3 **F-DBSTexC**($\mathcal{X}$,$\mathcal{Y}$, ϵ, $N_{\min_1}$, $N_{\min_2}$, $N_{\max_1}$, $N_{\max_2}$)

```
Input: $\mathcal{X}$,$\mathcal{Y}$, $\epsilon$, $N_{\min_1}$, $N_{\min_2}$, $N_{\max_1}$, $N_{\max_2}$
Output: Clusters with different labels $C$
Initialization: $C \leftarrow 0$; $n \leftarrow |\mathcal{X}|$; $m \leftarrow |\mathcal{Y}|$; $p_i$ is a point in
    the set $\mathcal{X}$
1: for each $p_i$ do
2:     if $p_i$ is not visited then
3:         Mark $p_i$ as visited
4:         $[\mathcal{X}_\epsilon(p_i), \mathcal{Y}_\epsilon(p_i)]$ = RangeQuery($p_i$)
5:         if $|\mathcal{X}_\epsilon(p_i)| \geq N_{\min_1}$ & $|\mathcal{Y}_\epsilon(p_i)| \leq N_{\max_2}$ then
6:             $C \leftarrow C + 1$
7:             ExpandCluster($p_i$, $\mathcal{X}_\epsilon(p_i)$, $\mathcal{Y}_\epsilon(p_i)$)
```

Algorithm 4 ExpandCluster(p_i, $\mathcal{X}_\epsilon(p_i)$, $\mathcal{Y}_\epsilon(p_i)$)

```
Input: $p_i$, $\mathcal{X}_\epsilon(p_i)$, $\mathcal{Y}_\epsilon(p_i)$
Output: Cluster $C$ with all of its members
1: Add $p_i$ to the current cluster with fuzzy score $\mu_{p_i}$
2: for each point $p_j$ in the set $\mathcal{X}_\epsilon(p_i)$ do
3:     if $p_j$ is not visited then
4:         Mark $p_j$ as visited
5:         $[\mathcal{X}_\epsilon(p_j), \mathcal{Y}_\epsilon(p_j)]$ = RangeQuery($p_j$)
6:         if $|\mathcal{X}_\epsilon(p_j)| \geq N_{\min_1}$ & $|\mathcal{Y}_\epsilon(p_j)| \leq N_{\max_2}$ then
7:             $\mathcal{X}_\epsilon(p_i) = \mathcal{X}_\epsilon(p_i) \cup \mathcal{X}_\epsilon(p_j)$
8:             $\mathcal{Y}_\epsilon(p_i) = \mathcal{Y}_\epsilon(p_i) \cup \mathcal{Y}_\epsilon(p_j)$.
9:             Add $p_j$ to the current cluster with fuzzy score $\mu_{p_j}$
10:    if $p_j$ does not have a label then
11:        Add $p_j$ to the current cluster
12: if $|\mathcal{Y}_\epsilon(p_i)| \neq 0$ then
13:     for each point $q_j$ in the set $\mathcal{Y}_\epsilon(p_i)$ do
14:         if $q_j$ is not visited then
15:             Mark $q_j$ as visited
16:             if $q_j$ does not have a label then
17:                 Add $q_j$ to the current cluster
```

B. EXPERIMENTAL EVALUATION

We summarize the experimental results in Table 5 according to different values of $\alpha \geq 0$. Similarly as in the original DBSTexC case, for the parameter set $(\epsilon, N_{\min_1}, N_{\min_2}, N_{\max_1}, N_{\max_2})$, we stepwise test the parameter combinations via exhaustive search. From the table, one can make the following insightful observations:

- The clustering quality of F-DBSTexC is higher than or at least equal to that of DBSTexC for all chosen POIs, showing the performance gain over DBSTexC by up to 27.33%.

[4] Other types of membership functions such as the exponential membership function [34] can also be applicable.

[5] To further improve the performance of **F-DBSTexC**, the cardinality μ_p can also be given in a different way (e.g., the max argument or the geometric mean).

TABLE 5. Experimental results for DBSTexC and F-DBSTexC .

	$A^{\alpha}\mathcal{F}_1$ ($\alpha = 0$)		
POI name	DBSTexC (X)	F-DBSTexC (Y)	Improvement Rate $\left(\frac{Y-X}{X} \times 100\%\right)$
Hyde Park	0.7391	0.7556	2.23
Regent's Park	0.7851	0.7949	1.25
University of Oxford	0.6930	0.7186	3.69
Edinburgh Castle	0.8364	0.8503	1.66
	$A^{\alpha}\mathcal{F}_1$ ($\alpha = 0.5$)		
Hyde Park	0.3058	0.3063	0.16
Regent's Park	0.3188	0.3325	4.30
University of Oxford	0.2062	0.2403	16.54
Edinburgh Castle	0.1741	0.1874	7.64
	$A^{\alpha}\mathcal{F}_1$ ($\alpha = 0.75$)		
Hyde Park	0.2284	0.2302	0.79
Regent's Park	0.2219	0.2228	0.41
University of Oxford	0.1673	0.1808	8.07
Edinburgh Castle	0.1510	0.1662	10.01
	$A^{\alpha}\mathcal{F}_1$ ($\alpha = 1$)		
Hyde Park	0.1816	0.1896	4.41
Regent's Park	0.1844	0.1848	0.22
University of Oxford	0.1288	0.1640	27.33
Edinburgh Castle	0.1412	0.1412	0

- Although F-DBSTexC has slightly better performance than that of DBSTexC for the two POIs located in London (i.e., Hyde Park and Regent's Park), it remarkably outperforms DBSTexC for POIs in smaller cities such as University of Oxford and Edinburgh Castle.

The first observation can be easily understood because F-DBSTexC is a fuzzy extension of DBSTexC; therefore, its performance is guaranteed to be at least as good as that of DBSTexC. On the other hand, the second observation may not be straightforward. We scrutinize the geographic distribution of tweets in various locations and notice that in general, POIs in crowded cities like London are surrounded by a significant number of POI-irrelevant tweets. As a result, further extension of the area covered by the clusters would not be beneficial. However, for POIs in smaller cities such as Oxford and Edinburgh, the geographic distribution of POI-irrelevant tweets around a POI tends to be much more sparse, enabling fuzzy extension of DBSTexC to work effectively. This remark highlights our proposition that F-DBSTexC is a dynamic extension of DBSTexC, allowing DBSTexC to apply in different situations with diverse types of POIs.

C. COMPUTATIONAL COMPLEXITY

Compared to DBSTexC, F-DBSTexC relaxes the constraints on a point's neighborhood density. However, the computational complexity of F-DBSTexC is still dominated by the function RangeQuery(), and F-DBSTexC invokes the function exactly once for every POI-relevant data point. Therefore, the computational complexity of F-DBSTexC is of the same order as that of DBSTexC, which is $\mathcal{O}(n^2+nm)$. More specifically, the complexity of F-DBSTexC is $\mathcal{O}(n^2)$ in a region where $m = c \times n$ for a constant c, and it follows $\mathcal{O}(n^{1+\beta})$ in another region where $m = n^{\beta}$ for $\beta > 1$.

VII. CONCLUDING REMARKS

As an extended version of DBSCAN, we introduced DBSTexC, a new spatial clustering algorithm that further leverages textual information on Twitter, composed of n POI-relevant tweets and m POI-irrelevant tweets. The algorithm is beneficial when we aim to find clusters from geotagged tweets which are heterogeneous in terms of textual description since DBSTexC effectively excludes regions containing a huge number of undesired POI-irrelevant tweets. The computational complexity of DBSTexC was shown to be $\mathcal{O}(n^2)$ in a region where $m = c \cdot n$ for a constant $c > 0$, and $\mathcal{O}(n^{1+\beta})$ in the other region where $m = n^{\beta}$ for $\beta > 1$. We demonstrated the performance of DBSTexC to be far superior to that of DBSCAN in terms of our performance metric $\bar{A}^{\alpha}\mathcal{F}_1$, where $\alpha \geq 0$ is the area exponent. As a further extension, we introduced F-DBSTexC, which incorporates the notion of fuzzy clustering into DBSTexC. By fully capturing their geographic features, the F-DBSTexC algorithm was shown to outperform the original DBSTexC for the POIs located especially in sparsely-populated cities. The design methodology that DBSTexC and F-DBSTexC provide takes an important step towards a better understanding of jointly utilizing spatial and textual information in designing density-based clustering and towards a broad range of applications from geo-marketing to location-based services such as geo-targeting, geo-fencing, and Beacons.

ACKNOWLEDGMENT

This paper was presented in part at the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, Sydney, Australia, July/August 2017. This paper has been significantly extended based on the prior work [41].

REFERENCES

[1] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*, 3rd ed. Amsterdam, The Netherlands: Elsevier, 2011.

[2] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Appl. Statist.*, vol. 28, no. 1, pp. 100–108, 1979.

[3] R. T. Ng and J. Han, "CLARANS: A method for clustering objects for spatial data mining," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 5, pp. 1003–1016, Sep. 2002.

[4] C. Fraley and A. E. Raftery, "Model-based clustering, discriminant analysis, and density estimation," *J. Amer. Statist. Assoc.*, vol. 97, no. 458, pp. 611–631, Jun. 2002.

[5] D. H. Fisher, "Improving inference through conceptual clustering," in *Proc. 6th Nat. Conf. Artif. Intell. (AAAI)*, Seattle, WA, USA, Jul. 1987, pp. 461–465.

[6] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. Hoboken, NJ, USA: Wiley, 1990.

[7] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, Montreal, QC, Canada, Jun. 1996, pp. 103–114.

[8] W. Wang, J. Yang, and R. R. Muntz, "STING: A statistical information grid approach to spatial data mining," in *Proc. 23rd Int. Conf. Very Large Data Bases (VLDB)*, Athens, Greece, Aug. 1997, pp. 186–195.

[9] R. Agrawal, J. E. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, Seattle, WA, USA, Jun. 1998, pp. 94–105.

[10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Data Mining Knowl. Discovery*, vol. 96, no. 34, pp. 226–231, Aug. 1996.

[11] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, Philadelphia, PA, USA, May/Jun. 1999, pp. 49–60.

[12] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications," *Data Mining Knowl. Discovery*, vol. 2, no. 2, pp. 169–194, Jun. 1998.

[13] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *WIREs Data Mining Knowl. Discovery*, vol. 1, no. 3, pp. 231–240, May/Jun. 2011.

[14] D. Birant and A. Kut, "ST-DBSCAN: An algorithm for clustering spatial–temporal data," *Data Knowl. Eng.*, vol. 60, no. 1, pp. 208–221, Jan. 2007.

[15] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining (PAKDD)*, Gold Coast, QLD, Australia, Apr. 2013, pp. 160–172.

[16] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?" in *Proc. 19th Int. Conf. World Wide Web (WWW)*, Raleigh, NC, USA, Apr. 2010, pp. 591–600.

[17] F. Morstatter, J. Pfeffer, H. Liu, and K. M. Carley, "Is the sample good enough? Comparing data from Twitter's streaming API with Twitter's Firehose," in *Proc. 7th Int. AAAI Conf. Weblogs Social Media (ICWSM)*, Boston, MA, USA, Jul. 2013, pp. 400–408.

[18] I. De Felipe, V. Hristidis, and N. Rishe, "Keyword search on spatial databases," in *Proc. 24th IEEE Int. Conf. Data Eng. (ICDE)*, Cancun, Mexico, Apr. 2008, pp. 656–665.

[19] G. Cong, C. S. Jensen, and D. Wu, "Efficient retrieval of the top-*k* most relevant spatial Web objects," *Proc. VLDB Endowment*, vol. 2, pp. 337–348, Aug. 2009.

[20] B. Yao, F. Li, M. Hadjieleftheriou, and K. Hou, "Approximate string search in spatial databases," in *Proc. 26th IEEE Int. Conf. Data Eng. (ICDE)*, Long Beach, CA, USA, Mar. 2010, pp. 545–556.

[21] Y. Tao and C. Sheng, "Fast nearest neighbor search with keywords," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 4, pp. 878–888, Apr. 2014.

[22] D.-W. Choi and C.-W. Chung, "A K-partitioning algorithm for clustering large-scale spatio-textual data," *Inf. Syst.*, vol. 64, pp. 1–11, Mar. 2017.

[23] D. Wu and C. S. Jensen, "A density-based approach to the retrieval of top-K spatial textual clusters," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, Indianapolis, IN, USA, Oct. 2016, pp. 2095–2100.

[24] D. D. Vu, H. To, W.-Y. Shin, and C. Shahabi, "GeoSocialBound: An efficient framework for estimating social POI boundaries using spatio–textual information," in *Proc. 3rd Int. ACM SIGMOD Worksh. Manag. Min. Enriched Geo-Spatial Data (GeoRich)*, San Francisco, CA, USA, Jun. 2016, Art. no. 3.

[25] W.-Y. Shin, B. C. Singh, J. Cho, and A. M. Everett, "A new understanding of friendships in space: Complex networks meet Twitter," *J. Inf. Sci.*, vol. 41, no. 6, pp. 751–764, 2015.

[26] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3336–3341, Mar. 2009.

[27] Y. van Gennip *et al.*, "Community detection using spectral clustering on sparse geosocial data," *SIAM J. Appl. Math.*, vol. 73, no. 1, pp. 67–83, Jan. 2013.

[28] B. Wang and X. Wang, "Spatial entropy-based clustering for mining data with spatial correlation," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining (PAKDD)*, Shenzhen, China, May 2011, pp. 196–208.

[29] J. C. Bezdek, W. Full, and R. Ehrlich, "FCM: The fuzzy c-means clustering algorithm," *Comput. & Geosci.*, vol. 10, nos. 2–3, pp. 191–203, 1984.

[30] S. Miyamoto, H. Ichihashi, and K. Honda, *Algorithms for Fuzzy Clustering*. Berlin, Germany: Springer, 2008.

[31] M. J. Li, M. K. Ng, Y.-M. Cheung, and J. Z. Huang, "Agglomerative fuzzy K-means clustering algorithm with selection of number of clusters," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 11, pp. 1519–1534, Nov. 2008.

[32] A. Smiti and Z. Eloudi, "Soft DBSCAN: Improving DBSCAN clustering method using fuzzy set theory," in *Proc. 6th Int. Conf. Human Syst. Interact. (HSI)*, Sopot, Poland, Jun. 2013, pp. 380–385.

[33] N. Zahid, O. Abouelala, M. Limouri, and A. Essaid, "Fuzzy clustering based on K-nearest-neighbours rule," *Fuzzy Sets Syst.*, vol. 120, no. 2, pp. 239–247, Jun. 2001.

[34] E. N. Nasibov and G. Ulutagay, "Robustness of density-based clustering methods with various neighborhood relations," *Fuzzy Sets Syst.*, vol. 160, no. 24, pp. 3601–3615, Dec. 2009.

[35] H.-P. Kriegel and M. Pfeifle, "Density-based clustering of uncertain data," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Chicago, IL, USA, Aug. 2005, pp. 672–677.

[36] D. Ienco and G. Bordogna, "Fuzzy extensions of the DBSCAN clustering algorithm," *Soft Comput.*, vol. 22, no. 5, pp. 1719–1730, Mar. 2018.

[37] G. Ulutagay and E. Nasibov, "Fuzzy and crisp clustering methods based on the neighborhood concept: A comprehensive review," *J. Intell. Fuzzy Syst., Appl. Eng. Technol.*, vol. 23, no. 6, pp. 271–281, Nov. 2012.

[38] M. Mathioudakis and N. Koudas, "TwitterMonitor: Trend detection over the Twitter stream," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, Indianapolis, IN, USA, Jun. 2010, pp. 1155–1158.

[39] S. Reis *et al.*, "UK gridded population based on Census 2011 and Land Cover Map 2007," NERC Environ. Inf. Data Centre, Tech. Rep., 2016.

[40] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN," *ACM Trans. Database Syst.*, vol. 42, no. 3, pp. 19:1–19:21, Aug. 2017.

[41] M. D. Nguyen and W.-Y. Shin, "DBSTexC: Density-based spatio-textual clustering on Twitter," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, Sydney, NSW, Australia, Jul./Aug. 2017, pp. 23–26.

MINH D. NGUYEN was born in Hanoi, Vietnam. He received the B.Eng. degree in mobile systems engineering from Dankook University, Yongin, South Korea, in 2018. He is currently pursuing the master's degree with the Electrical Engineering Department, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia. His research interests include social network analysis, data mining, and machine learning.

WON-YONG SHIN (S'02–M'08–SM'16) received the B.S. degree in electrical engineering from Yonsei University, Seoul, South Korea, in 2002, and the M.S. and Ph.D. degrees in electrical engineering and computer science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2004 and 2008, respectively.

From 2008 to 2009, he was with the Brain Korea Institute and CHiPS, KAIST, as a Postdoctoral Fellow. In 2009, he joined the School of Engineering of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA, as a Postdoctoral Fellow. He was promoted to Research Associate, in 2011. From 2012 to 2019, he was a Faculty Member with the Department of Computer Science and Engineering, Dankook University, Yongin, South Korea. Since 2019, he has been with the Department of Computational Science and Engineering, Yonsei University, where he is currently an Associate Professor. His research interests include information theory, communication, signal processing, mobile computing, big data analytics, and online social networks analysis. He served as an Organizing Committee Member of the 2015 IEEE Information Theory Workshop, the 2017/2018 International Conference on ICT Convergence, and the 2018 International Conference on Information Networking. He received the Bronze Prize of the Samsung Humantech Paper Contest, in 2008, and the KICS Haedong Young Scholar Award, in 2016. He has served as an Associate Editor for the *IEIE Transactions on Smart Processing and Computing* and the *Journal of Korea Information and Communications Society*. From 2014 to 2018, he served as an Associate Editor for the *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*. He has served as a Guest Editor for the *Energies*—Special Issue on Green Radio, Energy Harvesting, and Wireless-Powered Communications for Beyond-5G Wireless Systems, *The Scientific World Journal*—Special Issue on Challenges towards 5G Mobile and Wireless Communications, and the *International Journal of Distributed Sensor Networks*—Special Issue on Cloud Computing and Communication Protocols for IoT Applications.