IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 8, 2020, accepted February 26, 2020, date of publication March 3, 2020, date of current version March 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2977996

A Personalized and Approximated Spatial Keyword Query Approach

XIANGFU MENG, PAN LI, AND XIAOYAN ZHANG
School of Electronic and Information Engineering, Liaoning Technical University, Huludao 125105, China

Corresponding author: Xiangfu Meng (marxi@126.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772249, and in part by the General Research Foundation of Liaoning Education Department, China, under Grant LJ2019QL017.

ABSTRACT With the universal use of GPS and rapid increase of spatial Web objects, spatial keyword query has been widely used in Location-Based Services (LBS). Most of the existing spatial keyword query processing models only support location proximity and strict text matching which makes the semantically related objects cannot be provided to users and even may lead to the empty answer problem. In addition, the current index structures (such as IR-tree, Quadtree) cannot process numerical attributes which are usually contained in the descriptive information associated to the spatial objects. To deal with these problems, this paper proposes a spatial keyword query method that can support semantic approximate query processing. Firstly, the user original query is expanded by Conditional Generative Adversarial Nets (CGAN) method to generate a series of query keywords that are semantically related to the original query keywords. And then, a hybrid index structure called AIR-tree is built to facilitate the query matching, which can support the text semantic matching and process numerical attributes with Skyline method. Experimental analysis and results demonstrate that the proposed method achieves higher execution efficiency and better user satisfaction compared with the state-of-the-art methods.

INDEX TERMS Spatial keyword query, CGAN, AIR-tree, skyline.

I. INTRODUCTION

With the widespread use of the mobile Internet, more and more spatial Web objects are emerging on the Internet. A spatial object mainly contains the location information (usually represented by the latitude and longitude), text information (e.g., name, facilities, categories, etc.), and numerical information (e.g., price, user ratings, etc.). The Location-Based services (LBS), such as Ctrip, Didi, Foursquare, and Yelp, become more and more popular as the increasing of spatial objects, and the spatial keyword query [1]–[4] is an important supporting technology for LBS. However, the current spatial keyword query models usually confronted with the following problems. First, they mainly focus on retrieving spatial objects that are matching to the query keywords in terms of textual similarity, but fail to consider the semantic similarity. In fact, the spatial objects that are semantically related to but mismatched to the query keywords may also be accepted by users. Second, they treat numerical values contained in the descriptive information as text keywords, while the numerical values represent the different meanings compared to the text keywords and the method for processing the numerical information is also very different from that of the text matching processing.

For example, some LBS systems, such as Airbnb, TripAdvisor, hotels.com, Craigslist, Yelp, and Zillow, all have Boolean attributes, categorical attributes, and a large number of numerical attributes. However, in most cases, these numerical attributes are generally discretized and converted into categorical attributes and then be processed by the text matching processing method, which may result in the unsatisfactory of the user's query needs and preferences.

Let's look at an example. Figure 1 contains 9 spatial objects (represented by circles) and the corresponding location information, textual keywords, and numerical attribute values are listed in Table 1. For a given query q: (<34.2, −81.839>, <chicken, KFC>, <0.3, 0.2, 0.5>), it consists of three parts. The first part is the query location, the second is a set of query keywords, and the third is a set of weights specified by

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Asif Naeem.

TABLE 1. The property of spatial objects and the spatial keyword query in Figure 1.

POI	Location	Text document	Numerical attributes		
	(Latitude, Longitude)	Keywords	Noise	Price	Congestion
o_1	(33.3306902, -111.9785992)	pizza, steak	0.3	0.5	0.7
o_2	(41.1195346, -81.4756898)	chicken, McDonald's	0.2	0.6	0.4
o_3	(33.5249025, -112.1153098)	tea, coffee	0.3	0.4	0.5
o_4	(40.2916853, -80.1048999)	chicken, McDonald's	0.5	0.3	0.6
o_5	(33.3831468, -111.9647254)	shopping, market	0.8	0.7	0.9
o_6	(48.7272, 9.14795)	bar, beer, chicken	0.9	0.7	0.9
o_7	(40.6151022445, -80.0913487465)	chicken, McDonald's	0.3	0.3	0.5
o_8	(36.1974844, -115.2496601)	bread, sandwich	0.4	0.3	0.4
o_9	(36.20743, -115.26846)	movie, drink	0.2	0.4	0.3

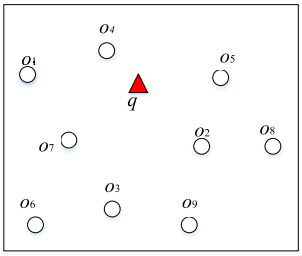

FIGURE 1. An example of spatial objects and spatial keyword query.

the user on the numerical attributes, which reflects the user's concern on the attributes, the larger the weight means the user cares more about the corresponding attribute. Note that, the values of numerical attribute of spatial object are different from the values of weights specified by the query. The former means the relevance degree of a spatial object to the corresponding attribute while the latter means the user's concern degree on the attribute. In general, the lower the values of numerical attributes means the better the scenario (e.g., low price, low noise, etc.). The numerical attribute values and the weights are both normalized into the unit interval [0, 1]. The query q is represented by a triangle in Figure 1. The purpose of q is to find the nearest ''KFC'' restaurant that provides ''chicken'' with features of ''low price'', ''low noise'', and ''uncrowded''. If we use a strict text matching model, there would be no answer object. However, ''KFC'' and ''McDonald's'' are similar to each other in semantics, so o_2, o_4 and o_7 can be taken into consideration as the results. Furthermore, for these three objects, o_4 is the closest to q in terms of location, o_7 is better than o_4 in terms of numerical attributes, o_7 is better than o_2 in terms of the price attribute, while o_2 is better than o_7 in terms of noise and congestion. As we know that the user cares more about the attribute *Congestion*, thus the query result should be o_2 rather than o_7. Therefore, it can be seen that the numerical information associated to the spatial object is also crucial for the spatial keyword query and cannot be treated as simple text keywords. Inspired by the above observations, the purpose of this paper is to establish a spatial keyword query processing model and to build an effective hybrid index structure to improve query efficiency. This model can integrate the location proximity, text similarity, semantic approximation, and user's satisfaction on numerical attributes between the spatial objects and query to evaluate the query results.

The main contributions of this paper are summerized as follows:

(i) We propose a CGAN-based method to expand the original query keywords to a series of semantically related keywords that are used to obtain the semantic approximate query results, even if the query keywords are very rare.

(ii) We present a Skyline-based numerical attribute processing method which can efficiently deal with the numerical attribute values associated to spatial objects and make the query results satisfy the user's personalized needs more closely.

(iii) A new hybrid index structure AIR-tree is constructed. This index structure can directly obtain the Skyline set of the corresponding numerical attributes of the intermediate nodes, and can integrally index the location information, text information and semantic information.

(iv) Comprehensive experiments are conducted over the real Point of Interest (POI) datasets to demonstrate the efficiency and effectiveness of our proposed spatial keyword query model and hybrid index.

The structure of the rest paper is organized as follows. Section 2 briefly reviews the related work. Section 3 defines the problem and presents the solution framework. Section 4 proposes the spatial keyword approximate query approach while Section 5 describes the AIR-tree index structure and presents the corresponding algorithms. Experiments are conducted in Section 6 and the paper is concluded in Section 7.

II. RELATED WORK

The existing spatial keyword query processing models mainly include Boolean range query, Boolean k nearest neighbor (kNN) query [5]–[7], top-k range query [8], [9], and top-k k nearest neighbor query [3], [10], [11] according to the literature [12]. The basic idea of these approaches is to construct a result scoring function according to the text similarity and location proximity between spatial objects and spatial keyword queries. And then, the text and spatial hybrid index technology is used to improve query efficiency. The disadvantage of the Boolean range query is that it cannot control the scale of query results and does not rank the query results. Boolean k nearest neighbor query ranks the query results by the distance between the spatial objects and the query point. Top-k range query finds the k spatial objects having the highest textual

relevance to the query keywords and their locations are within the query region. Top-k nearest neighbor query ranks the top-k spatial objects according to their location proximity and text relevancy. More specifically, it retrieves the k objects having the highest ranking scores which are measured by a weighted combination of their distances to the query location and the textual similarity between their textual descriptions and query keywords. The first two methods need the text description of spatial objects containing all query keywords, which may lead to the few/no answer problem, or the query results are far away from the query point. The latter two queries do not require the descriptive information of spatial objects to contain all the query keywords, and the spatial objects containing only part of the query keywords can also be treated as the query results. However, top-k range query ranking method only considers the text relevance of spatial objects but ignores the location proximity. The top-k nearest neighbor query considers both the location proximity and text relevance between spatial objects and query. Collective Spatial Keyword Query (CSKQ) [13], [14] returns a set of objects that collectively cover user's query keywords, those objects are close to the query location and have small inter-object distances. Following the CSKQ, the Reverse Collective Spatial Keyword Query (RCSKQ) [15], [16] returns a region, in which the query objects are qualified objects with the highest spatial and textual similarity. Recently, the top-k kNN, CSKQ, and RCSKQ query models are the most popular techniques in the current spatial keyword query processing, and the CSKQ and RCSKQ are the variants of the top-k nearest neighbor. However, it should be pointed out that the top-k kNN query and its variants rarely consider the relevancy of query keywords and text documents of spatial objects in semantics. Furthermore, the top-k objects in the answer set are usually very similar to each other which can neither effectively reflect the features of the entire dataset nor broaden the user's perspectives.

To quickly retrieve the matching query results, some hybrid index structures are developed to assist the online query processing. The existing hybrid indexing technologies for spatial keyword query are mainly include IR-tree [10], [11], [17], IR^2-tree [5], R*-Tree [18], QuadTree [2], S2I [9] and other spatial index structures. R-tree [19] is the most basic spatial index structure, most of the other spatial index structures are its variants. The text indexing techniques mainly contain the inverted files, signature files, and bitmaps and they are focus on the exact text matching for query keywords, which may result in too few or no results due to the diversity of text expressions. In response to the above problems, related studies such as literature [20]–[23] proposed a series of indexing techniques to deal with spelling errors. However, these methods did not take the semantic similarity/relevance between texts into account. Although a few number of recent works have studied the semantic matching of spatial keyword queries [24], spatial objects include not only location information and text information, but also numerical information such as price and user rating. As a hybrid index structure composed of clustering layer and spatial layer, QDR-Tree [25] needs to convert keywords into bitmap first, and then uses search scaling to achieve similar keyword matching. Such query processing would lead to a low query efficiency and cannot achieve the goal of semantic approximation query.

Our work is also relevant to the text semantic similarity measuring. Text semantic similarity measuring methods can be mainly classified into the following three categories: (i) KB (Knowledge based)-based similarity measure. Text semantic similarity can be estimated by defining topological similarity by using ontology to define the distance between terms/concepts. The methods based on KB, such as WordNet, Probase, and Wikipedia, were used to split text and then capture the keyword relationships [26]. However, the keywords and their relationship measures in WordNet and Wikipedia are subjective and cannot reflect the relationships between keywords against the datasets. In addition, the keywords or concepts uncovered by KB would not be processed. (ii) The topic model-based similarity measure. The probabilistic topic model can be used to approximate the semantic processing of text information, which is a statistical method to analyze the keywords in a document and discover the topic of these keywords and the relationships between these topics. There have been a number of classic thematic Models, such as Latent Dirichlet Allocation (LDA) [27], Dynamic Topic Model [28], Dynamic HDP [29], Sequential Topic Models [30]. Topic model is widely used in text classification, user behavior analysis, functional area discovery, etc. A lot of work has been done to apply the topic model on location-based service recommendation system [31], [32]. Although the topic models have achieved a certain improvement against the traditional similarity measuring methods such as Bag of Words (BOW) and CVM-VSM models [33], the significance of improvement and generalized ability is not enough in processing the special scenarios (such as short texts). Unfortunately, the text description of spatial objects is often short text, as [34] pointed out, the short texts usually do not contain sufficient statistical information to support traditional topic models for text processing. (iii) The word-embedding-based similarity measure. Word-embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to dense, distributed, fixed-length vector representations in a low-dimensional space relative to the vocabulary size. The popular techniques of word-embedding mainly contain the Word2Vec [35], [36] (such as Skip-gram and CBOW), genism, FastText, and GloVe [37]. Word-embedding technique is very successful in NLP. However, word-embedding techniques such as Word2Vec supposes the nearby/adjacent words/phrases (in a fixed window size) usually having the strong contextual relations while it cannot deal with the rare query and the correlations between keywords cannot be measured accurately. (iv) Conditional Generative Adversarial Nets (CGAN [38]) based similarity measure. The CGAN was used to generate bid keywords directly from query in sponsored search ads selection,

especially for rare queries in [39]. In the query expansion stage, based on the user query, the sequence-to-sequence model is used as the generator to generate keywords, and then the recurrent neural network (RNN) model is used as the discriminator to play a game against the generator. By training the generator, the keywords that are semantically related to the initial query keywords can be generated directly, and thus the original query can be expanded by these generated keywords. Inspired by [39], we take advantages of CGAN in our scenario to expand the query keywords in the spatial keyword query.

To the best of our knowledge, there is no related work considering the integrated similarities between spatial objects and query in terms of location proximity, text similarity, semantic relevance, and user's satisfaction on numerical attributes at the same time, and thus there is no hybrid spatial index structure that supports the above integrated query. The aim of this paper is to establish a spatial keyword query processing model and then proposes a corresponding effective hybrid index structure to improve the query efficiency. The model we established can integrate the location proximity, text semantic similarity, and user's satisfaction on numerical attributes between query and spatial objects into a query result scoring function.

III. PROBLEM DEFINITION AND SOLUTION

In this section, we first define the spatial keyword query problem and then present the solution framework.

A. PROBLEM DEFINITION

Given a spatial dataset $O = \{o_1, o_2, \ldots, o_n\}$, each spatial object $o_i \in O$ is represented by a tuple (λ, K, A), where $o_i.\lambda$ is the location information of o_i, $o_i.K$ is a set of text keywords associated to o_i, and $o_i.A$ is a set of numerical attributes of o_i. The value $o.a_i \in o_i.A$ is normalized into the unit interval [0, 1]. We assume that the lower the numerical attribute value means the closer the scenario to the user preferences (such as the low noise and low price). On the contrary, if the higher value is better for some cases (e.g., ambient atmosphere, rating, etc.), the value of a_i should be converted to $a_i = 1 - a_i$.

The spatial keyword query q is represented by a tuple (λ, K, W), where $q.\lambda$ is the location information of q, $q.K$ is a set of query keywords, $q.W$ is a set of weights corresponding to different numeric attributes, which is specified by the user according to his/her preferences on the numerical attributes. The larger the value of weight means the user cares more about the corresponding attribute. For example, the weight specified on the attribute *Congestion* is 0.5 in Introduction means the user is more concerned about *Congestion* compared with *Noise* and *Price*. Note that, $\forall q.w \in q.W$, $q.w \geq 0 (i = 1, \ldots, |q.W|)$, and $\sum_{i=1}^{|q.W|} q.W = 1$.

B. SOLUTION FRAMEWORK

The solution proposed in this paper is shown in Figure 2, which can be divided into offline pre-processing stage and online processing stage. The construction of AIR-tree hybrid index structure and the calculation of semantic similarity are completed in the offline stage while the query result computation and top-k result retrieval are processed during the online stage. The online spatial keyword query processing stage consists of the following two steps,

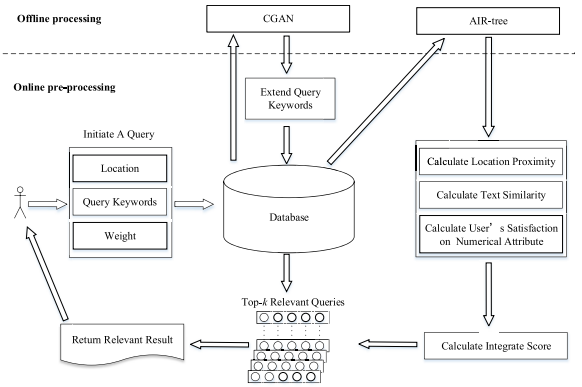

FIGURE 2. Solution framework.

(i) For the user initial query $q = (\lambda, K, W)$, the original query keywords are first expanded to be a set of semantically related keywords by using CGAN. After this, the original query $q = (\lambda, K, W)$ is expanded/relaxed to be $q = (\lambda, \tilde{K}, W)$ with the keywords semantically related to the original query, where $q.\tilde{K}$ is expanded/relaxed from $q.K$. Since the generator of CGAN has been trained in the offline stage, the relevant keywords can be directly generated by CGAN, and thus the online query processing efficiency can be significantly improved.

(ii) For the expanded/relaxed spatial keyword query $q = (\lambda, \tilde{K}, W)$, AIR-tree which is built during the offline stage is leveraged to retrieve the top-k spatial objects having the high location proximity, text semantic similarity, and user's satisfaction on numerical attributes to it.

IV. SPATIAL KEYWORD APPROXIMATE QUERY APPROACH

In this section, we first propose the original query expansion method by using CGAN, and then describe the measuring methods for location proximity, text semantic similarity, and user's satisfaction on numerical attributes between spatial objects and query, respectively.

A. QUERY EXPANSION

To realize the spatial keyword approximate query, it should first expand the original query keywords with the semantically related keywords. In this section, we propose a CGAN-based method to expand the original query. It can generated the semantically related keywords directly from a given query keyword even if the query keyword is rare. Generative Adversarial Nets (GAN [40]), whose basic idea is derived from the game theory of two-player game (that is, the sum of interests of two players is zero, and the gain of one party is the loss of the other), sets the players as a Generator and a Discriminator respectively and is trained by adversarial learning to estimate

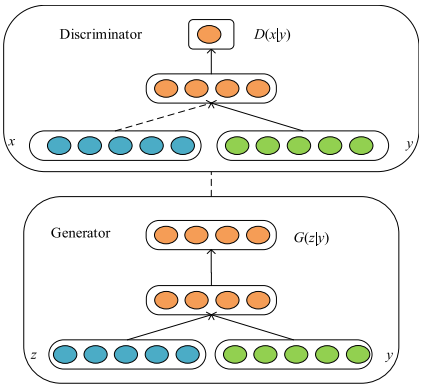

FIGURE 3. The structure and processing procedure of CGAN.

the potential distribution of data samples and generate new data samples. The purpose of the generator is to learn the real data distribution as much as possible, while the purpose of the discriminator is to distinguish whether the input data is from the real data or the generator as accurately as possible. To win the game, the two players need to constantly optimize and improve themselves.

Any differentiable function can be utilized to represent the generator and discriminator of GAN. In this paper, we use differentiable functions D and G to represent the discriminator and the generator respectively. Their inputs are real data x and random variable z, respectively. $G(z)$ is a sample generated by G that obeys the distribution of real data as much as possible. If the input of the discriminator is from real data, the label is 1. If the input sample is $G(z)$, the label is 0. Here the goal of D is to realize the two-class discrimination of data sources: true (from the distribution of real data x) or false (from the generator's pseudo data $G(z)$). The goal of G is to make the performance of the generated pseudo data $G(z)$ on $D(G(z))$ coincide with that of the real data x on $D(x)$. These two processes of adversarial and iterative optimization make the performance of D and G continuously improve. When the discriminating ability of D is improved to a certain extent and the source of data cannot be correctly identified, it can be considered that the generator G has learned the distribution of real data. G and D are both trained simultaneously, as if they are following the two-player min-max game with value function $V(G, D)$:

$$\min_G \max_D V(G,D) = E_{x\sim p_{data(x)}}[\log D(x)] + E_{z\sim p_{z(z)}}[\log(1 - D(G(z)))] \quad (1)$$

where, p_g is the generator's distribution over data x, $p_z(z)$ is a prior distribution of z, θ_g and θ_d contains the parameters of the generator and the discriminator, respectively. $G(x;\theta_g)$ is a mapping to data space, $D(x;\theta_d)$ is a second mutilayer perceptron that outputs a single scalar.

Conditional Generative Adversarial Nets (CGAN [38]) is expanded by adding some extra information y to GAN, where y can be any kind of additional information, such as category labels or data from other models. The structure of CGAN is shown in Figure 3. We can perform the conditioning by feeding y into both the discriminator and generator as additional input layer. In the generator, the prior input noise $p_z(z)$ and y are combined in joint hidden representation, and the adversarial training framework allows for considerable flexibility in how this hidden representation is composed. In the discriminator, x and y are presented as inputs added to a discriminative function. The objective function of CGAN can be defined as follows:

$$\min_G \max_D V(G,D) = E_{x\sim p_{data(x)}}[\log D(x|y)] + E_{z\sim p_{z(z)}}[\log(1 - D(G(z|y)))] \quad (2)$$

Inspired by the QE-CGAN framework which is proposed in literature [39] to generate bid keywords directly from query in sponsored search ads selection, we use a CGAN-based method to generate a set of keywords that are semantically related to the original query keywords. The CGAN-based method can capture the implicit or latent correlations between keywords that occurred very few times in the learning samples (such as query history and text documents), while the existing similarity measuring methods (e.g., TFIDF, PMI, LDA, etc) are lack for discovering such relationships between the low frequent occurrence keywords due to their statistic computation nature. In the query extension (query-keyword matching) stage, based on the user query, the sequence-to-sequence model is used as the generator to generate keywords, and then the RNN model is used as the discriminator to play a game against the generator. The policy gradient [41] is used to train the model. After training, given a user query, the generator can use different noise vectors that match many queries to generate a set of keywords semantically related to the original query keywords. That is, the semantic related keywords can be generated directly from a given query by training the generator, which can efficiently improve the online query expansion performance. The framework of CGAN-based method we proposed is shown in Figure 4.

The framework consists of a generator G and a discriminator D. Generator $G(z|q;\theta_g)$ is a sequence-to-sequence model, which is a bi-directional Gated Recurrent Unit (GRU [42]) parameterized by θ_g. Discriminator $D(k|q;\theta_d)$ is a parallelized RNN model, which is a one-directional GRU parameterized by θ_d. The objective function of CGAN-based method, which reflects the adversarial game between G and D, is written as follows:

$$\min_G \max_D V(G,D) = E_{(q,k)\sim p_{data(q,k)}}[\log D(k|q)] + E_{z\sim p_{z(z)}}[\log(1 - D(G(z|q)|q))] \quad (3)$$

For the given spatial keyword query in the Introduction, the original query keywords <chicken, KFC> is expanded to be the <chicken, KFC, McDonald's> by using the CGAN-based method.

To further demonstrate the reasonability and superiority of CGAN-based query expansion method, we compare it with the well-known Word2Vec-based word embedding method which is also effective for measuring the semantic relevancy between terms. Table 2 shows an example of the semantic

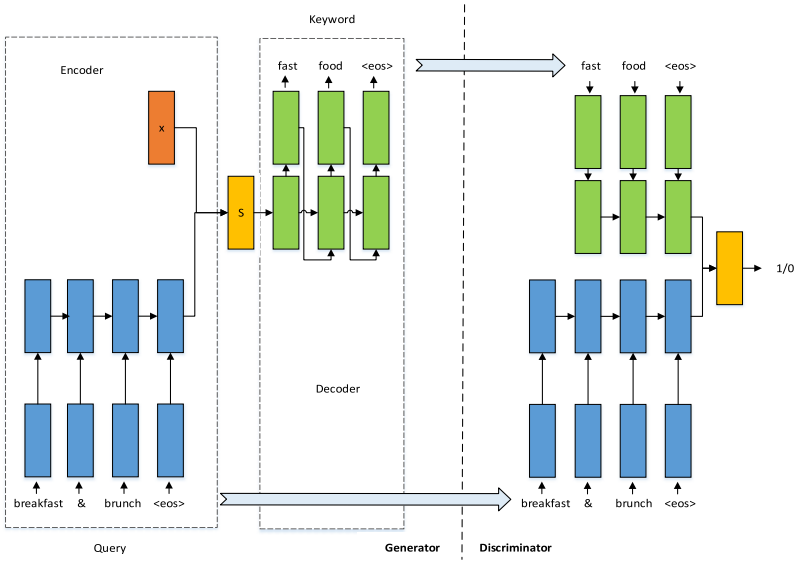

FIGURE 4. The framework of CGAN-based method.

TABLE 2. The keywords generated by Word2Vec-based word embedding method vs. CGAN-based query expansion/relaxation method.

Input query keyword	Output keywords by Word-Embedding method	Output keywords by CGAN-based method
sunset club	shopping, beauty & spas, home services, health & medical, teakwoods tavern & grill	leisure centers, hostels, climbing, golf galaxy, specialty roofing
home services	health & medical, automotive, beauty & spa, home service, health & medical	local services, active life, real estate, auto repair, event planning & services, arts & entertainment, hotels & travel
beauty & spas	las vegas, food, phoenix, home services, nightlife	health & medical, hair salons, nail salons, hair removal, skin care, local services
tea rooms	beauty & shopping, beauty & spas, home services, local services, active life	festivals, chicken shop, pet stores, internet cafes, street vendors, candy stores
coffee & tea	shopping, beauty & spas, home services, health & medical, local services	breakfast & brunch, bakeries, specialty food, burgers, sandwiches, desserts, fast food
breakfast & brunch	home services, health & medical, active life, hair salons, home & garden	burgers, pizza, fast food, sandwiches, coffee & tea
real estate agents	wine & spirits, nightlife, bars, real estate, restaurants, caterers	plumbing, home cleaning, sewing & alterations, discount store, laundry services, carpet cleaning, landscaping, public services & government, books
shoe repair	home services, social clubs, goodlife fitness	furniture reupholstery, landscape architects, screen printing/t-shirt printing, reiki, cvs pharmacy, landmarks & historical buildings, obstetricians & gynecologists

related keywords generated by using CGAN-based method (resp. Word2Vec-based word embedding method) for the given input keywords. It can be seen that the expanded keywords generated by CGAN are more reasonable for the given input keywords than that of Word2Vec-based method. For example, the generated keywords "health & medical, hair salons, nail salons, hair removal, skin care, local services" are very close to the corresponding given keyword "beauty & spas" in terms of semantics. This is because Word2Vec ignores the relationship between those phrases that appear less frequently and highly depends on the sliding window. When the sliding window is not large enough, the correlation between the words at the beginning and the end of the sentence would be ignored. Besides, due to the low volume of rare queries, it is difficult to accumulate enough terms to use statistical methods to achieve query keyword matching. In contrast, as we described above CGAN can effectively deal with the rare query and directly generate the relevant keywords because of its working theory.

B. QUERY RESULT EVALUATION

For a given spatial keyword query $q = (\lambda, K, W)$, we first expand the query keywords $q.K$ to be the expanded/relaxed query keywords $q.\tilde{K}$ by using CGAN-based method, and then we evaluate the query results to the expanded/relaxed query $q = (\lambda, \tilde{K}, W)$ from location proximity, text semantic similarity, and user's satisfaction on numerical attributes, respectively.

1) LOCATION PROXIMITY

Given a query q and a spatial object o, the location proximity between q and o can be computed as follows:

$$S_{spatial}(q, o) = 1 - \frac{dist(q.\lambda, o.\lambda)}{MaxDist} \tag{4}$$

TABLE 3. The location proximity and text semantic similarity between the query q and spatial objects.

Object	Location proximity	Text semantic similarity
o_1	0.6728	0.0000
o_2	0.9248	0.4082
o_3	0.6713	0.0000
o_4	0.9313	0.4082
o_5	0.6729	0.0000
o_6	0.0000	0.3333
o_7	0.9278	0.4082
o_8	0.6367	0.0000
o_9	0.6365	0.0000

where, $dist(q.\lambda, o.\lambda)$ is the Euclidean distance between o and q, $MaxDist$ is the maximum distance in the dataset D of all spatial objects. The Euclidean distance between objects o_i and o_j is generally calculated as $dist(o_i, o_j) = \sum_{k=1}^{m} d(o_i^{(k)}, o_j^{(k)})$, where m is the spatial dimensions of the spatial objects.

2) TEXT SEMANTIC SIMILARITY

We propose a two-step method for measuring the text semantic similarity between the set of text keywords $o.K$ associated to a spatial object o and the set of query keywords $q.K$ of q. The first step is to transform $o.K$ and the set of keywords $q.\tilde{K}$ (which is expanded/relaxed from $q.K$) into the vectors, which are represented by $V_{o.K}$ and $V_{q.\tilde{K}}$ respectively. Note that, we suppose M be the set of distinct keywords contained in the text information associated to all spatial objects in O and $q.\tilde{K}$. We also assume m is the number of keywords in M, that is, $m = |M|$. We let Δ be an fixed order on the keywords appearing in M. $M[i]$ refers to the $(i+1)$-th keyword of M based on the order Δ, where $i = \{0, \ldots, m-1\}$. If $M[i]$ appears among the keywords of document then $M[i] = 1$, otherwise it is 0. The second step leverages the Cosine similarity method to calculate the similarity between $V_{o.K}$ and $V_{q.\tilde{K}}$. The calculation method is defined as follows:

$$S_{text}(q, o) = \frac{\sum_{i=1}^{n} V_{o.K}[i] \cdot V_{q.\tilde{K}}[i]}{\sqrt{\sum_{i=1}^{n}(V_{o.K}[i])^2} \cdot \sqrt{\sum_{i=1}^{n}(V_{q.\tilde{K}}[i])^2}} \quad (5)$$

Table 3 shows the location proximity and text semantic similarity between the query q shown in Introduction and the spatial objects $o_i (i = 1, \ldots, 9)$ (listed in Table 1) calculated by using our location proximity and text semantic similarity measuring methods.

Clearly, the candidate query results would contain the semantically related spatial objects to the original query since the query keywords of the original query has been expanded by adding the relevant keywords.

3) USER'S SATISFACTION ON NUMERICAL ATTRIBUTE

To measure the user's satisfaction on numerical attributes between the query q and a spatial object o, this paper proposes a skyline-based method.

Given a set of tuples, the basic idea of Skyline is to calculate the dominance relationship between them. The Skyline set is a collection of tuples that are not subject to any other tuples in the dataset [43]. If q is better than p in at least one dimension and is not worse than p in all other dimensions, then q is said to dominate p. Furthermore, if a pair of tuples p and q do not dominate each other, then both tuples p and q should be in Skyline. For example, the low prices, high levels, and more parking positions are the good choice if a user looking for a resort considers *price*, *hotel level*, and the number of *parking positions*. Therefore, if p is in Skyline, there is no other q that is not in Skyline with lower price, higher level and more parking positions than p. Clearly, Skyline method has great advantages in finding good query results. This paper will take advantage of the Skyline query method to find the query results in order to satisfy the user preferences on numerical attributes more closely.

Suppose the relation D has n tuples with m attributes $\mathbf{A} = \{A_1, A_2, \ldots, A_m\}$, $t[A_i]$ is the value of the tuple t on the attribute A_i. Assume that for each attribute, the value in the dominance relationship has a total ordering of preferences (eg., $a > b$ indicates that a is better than b). A tuple $t \in D$ dominates another tuple $t' \in D$, denoted by $t \prec t'$, if and only if $\forall A \in \mathbf{A}, t[A] \geq t'[A]$ and $\exists A \in \mathbf{A}, t[A] \prec t'[A]$. In addition, if a tuple $t \in D$ is incomparable to another tuple $t' \in D$, it is represented as $t \sim t'$ if and only if t' and t do not domainate each other.

Skyline S is a collection of tuples in D that are not dominated by other tuples. For the objects o_2, o_4, o_7 in Figure 1, their numerical properties can be represented as $\{0.2, 0.6, 0.4\}$, $\{0.5, 0.3, 0.6\}$, $\{0.3, 0.3, 0.5\}$, respectively. It is clear to see that the first attribute and the third attribute of o_7 are better than o_4, and the second attribute of o_7 is equal to the second attribute of o_4. Thus, o_4 is dominated by o_7. Comparing o_2 and o_7, we can see that the first attribute and the third attribute of o_2 are better than o_7 while the second attribute of o_2 is inferior to o_7, which leads to o_2 and o_7 are incomparable. Thus, o_2 and o_7 should be both added into S.

After this, the user's satisfaction on numerical attributes between the spatial object o and query q can be computed as follows:

$$S_2(q, o) = 1 - \sum_{i=0}^{|q.W_i|} (q.W_i \cdot o.a_i) \quad (6)$$

For instance, if a user specifies $q.W_1$ to be 0.3, $q.W_2$ to be 0.2 and $q.W_3$ to be 0.5, then $S_2(q, o_2) = 1-(0.3*0.2+0.2*0.6+0.5*0.4) = 0.62$, $S_2(q, o_7) = 1-(0.3*0.3+0.2*0.3+0.5*0.5) = 0.6$. Clearly, the best answer is o_2. In contrast, if another user specifies $q.W_1$ to be 0.53, $q.W_2$ to be 0.37 and $q.W_3$ to be 0.1, then $S_2(q, o_2) = 0.632$, $S_2(q, o_7) = 0.68$, and thus the best answer would be o_7.

4) INTEGRATED QUERY RESULT SCORING FUNCTION

Based on the location proximity and text semantic similarity between the spatial object and query and the user satisfaction on numerical attributes as well, we can build the integrated scoring function for a set of spatial objects and the query q as

follows:

$$score(q, o) = \beta \cdot S_1 + (1 - \beta) \cdot S_2 \quad (7)$$

where β is an adjustment parameter, and its value is set to 0.7 (the discussion will be shown in Section 6 in detail). The factor S_1 is shown as follows:

$$S_1(q, o) = \alpha \cdot S_{spatial}(q, o) + (1 - \alpha) \cdot S_{text}(q, o) \quad (8)$$

where α is an adjustment parameter which is usually set to 0.5 in past researches. In this paper, we also follow this setting for the consistency.

V. HYBRID INDEX AND QUERY ALGORITHM

In this section, we propose the hybrid index structure and present the corresponding implementation algorithms.

A. HYBRID INDEX STRUCTURE

To facilitate the query matching procedure by simultaneously considering the location proximity, text semantic similarity, and user's satisfaction on numerical attributes, we propose a hybrid index structure which consists of the semantic layer and AIR-tree index layer.

1) THE SEMANTIC LAYER

CGAN is leveraged to expand the original query keywords. To make the expansion as good as possible, we used the same model settings as in [39]. For the generator of CGAN, we used a two-layer bi-directional GRU with the hidden size of 500. We also applied a dropout rate of 0.1 as regularization which helps reduce overfitting, that is, reducing performance on the training set, but improving performance on the testing set. During the decoding phase, we connected our network to a V-dimensional Softmax layer, where V is the overall vocabulary size for queries and keywords. We only keep tokens that appear more than 50 times to obtain a final dictionary size of 30,000 in order to reduce computation. The discriminator which is a single layer one-directional GRU with the hidden size of 150. As shown in Figure 4, the query q and the keyword k are a sequence of tokens $q_1, q_2, \ldots, q_n$ and $k_1, k_2, \ldots, k_{n'}$, respectively. In the encoding process, the standard RNN encoder model computes the thought vector by iterating $h_t = \sigma_h(W_h q_t + U_h h_{t-1})$, where h_t is the hidden layer vector, W_h and U_h are parameter matrices, and σ_h is the activation function. Note that, the input is a noise vector z conditioned on the query q, we compute the thought vector S using $S = \sigma_z(U_h h_t + U_z z)$ after parallelly connecting z and q, where U_z is the parameter matrix and σ_z is the activation function. In the decoding process, the RNN decoder model computes the conditional probability $p(k_1, k_2, \ldots, k_{n'}|z, q_1, q_2, \ldots, q_n)$ with a standard language model, in which the initial hidden state is set to the thought vector S. For the discriminator, we use a parallelized RNN model. In this model, query is fed into one RNN structure and keyword is fed into another RNN structure as shown in Figure 4. The thought vectors of these two RNNs are then fully connected to a hidden layer, and the hidden layer produces the final prediction (1/0) output. Pre-training has shown some benefits for the training process [44]. Therefore, we first pre-trained our sequence-to-sequence generator model for 2 epochs by the Adam optimizer [45] with the learning rate of 1e-4. Besides, the pre-training also helped us produce the initial word embedding for the follow-up GAN training. Furthermore, after the pre-training process, we would freeze the word embedding weights as in [39].

2) AIR-TREE INDEX LAYER

We build a new hybrid index structure (AIR-tree), which adds an *AttrFile* file to each node (including intermediate node and leaf node) based on IR-tree. Each node of AIR-tree records the spatial information of all objects in the sub-tree rooted at the current node, the text information summary (the set of keywords extracted from the text document associated to the node), numerical attribute information and pointers.

As shown in Figure 5, the information associated to each node of AIR-tree is divided into three parts: the first two parts are two pointers, the first of which points to the inverted file (*InvFile*) containing all the keywords associated to the node and the second of which points to the numerical attribute file (*AttrFile*). The third part is the collection of entities in the node (i.e., Entries in R-tree).

Each intermediate/non-leaf node and leaf node may contain multiple entries. For a leaf node, each of its entries consists of a quad, in the form of $<O, Rect, O.tid, O.aid>$, where O represents a spatial object, $Rect$ represents the minimum bounding rectangle (MBR) of the object, $O.tid$ is the textual information identifier of the object, and $O.aid$ is the identifier of the numerical attribute tuple of the object. For non-leaf nodes, each item among them is also composed of a quad which is represented as $<pN, Rect, N.pid, N.aid>$, where pN is the address of the child node N in the node, and $Rect$ refers to the MBR of all child nodes of the node, $N.pid$ is the document identifier of the node (the document contains the information of all the child nodes under the node), and $N.aid$ is the numerical attribute identifier of the node which contains the Skyline generated over the numerical attribute tuples contained in its child nodes.

During the online query processing stage, the AIR-tree index structure is leveraged by Algorithm 2 to perform the top-k query over the spatial dataset by simultaneously considering the location information, text information, and numerical attributes associated to the spatial objects. Firstly, the location proximity and text semantic similarity between the query and spatial objects is calculated by Equation (7). Secondly, the user's satisfaction on numerical attributes and the integrated ranking score are calculated. Lastly, the query results are sorted according to the integrated ranking score in descending order.

B. ALGORITHM

This section presents two algorithms, the Algorithm 1 is used for generating Skyline collection over the numerical

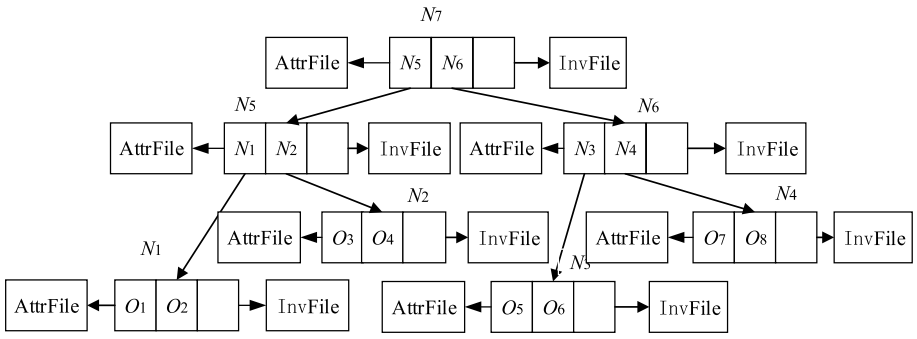

FIGURE 5. AIR-tree index structure.

Algorithm 1 The Algorithm for Generating Skyline Collection for Numerical Attribute Tuples

Input: numerical attribute tuple list *listattr*
Output: Skyline of *listattr*

```
sort listattr by the first attribute in ascending order.
for each tuple in listattr do
    let i = 0
    while i < tuple.size() do
        /* compare tuple[i] to the i-th element of other
        tuple tuple1[i] in listattr*/
        if tuple[i] ≤ tuple1[i] then
            listattr.remove(tuple1)/*remove other tuple
            tuple1 from listattr*/
            i += 1
        else
            listattr.remove(tuple)
return Skyline of listattr
```

attribute tuples associated to the spatial objects and the Algorithm 2 aims to find the top-k results by using the AIR-tree index.

Algorithm 1 is used in the process of building the AIR-tree. The *AttrFile* corresponding to the intermediate node of AIR-tree is the Skyline collection of numerical attribute tuples of its child nodes. For a given numerical attribute tuple list *listattr*, we first sort it by the value of the first attribute in ascending order. For a given tuple, if its value on each attribute is larger than the other tuples, it is removed from the *listattr* until such tuples do not exist. The Skyline can support the personalized querying according to user preferences and ultimately provide data items independent of other data items. During the query processing, the score S_2 of the numerical properties of each intermediate node is determined by its Skyline and user-specified weights, which determines which branch to be chosen to speed up the lookup.

The time complexity of the Algorithm 1 is $O(mnlog(n))$, where $m = tuple.size()$, $n = listattr.size()$. It is acceptable because the AIR-tree is built in the offline pre-processing stage and it can be updated periodically.

Algorithm 2 The Algorithm of Top-k Query by Using AIR-Tree

Input: Dataset O, expanded/relaxed query $q(\lambda, \tilde{K}, W)$, the number of results k, adjusting parameters α, β, hybrid index structure AIR-tree
Output: Candidate set list *result*.

```
result ← ∅, max heap heap ← ∅, score = 0
heap.add(root)/*root is the root of AIR-tree*/
while heap ≠ ∅ and result.size()<k do
    N=heap.poll()
    if N is an object then
        S1(q, N) = α · S_spatial(q, N) + (1 − α) · S_text(q, N)
        S2(q, N) = 1 − Σ_{i=0}^{|q.Wi|} (q.Wi · N.ai)
        score(q, N) = β · S1 + (1 − β) · S2.
        result.add(N)
    else
        for entry e in N do
            heapEntry.add(e)
            if q.K̃ == heapEntry.getId().getKeyword()
            then
                calculates the S1 of the coupling
                correlation between the location
                information and the text information by
                Equation (8).
                calculates the S2 by using Equation (6)
                after obtaining the value of numerical
                attributes by heapEntry.getId().getAttr().
                score = β · S1 + (1 − β) · S2.
                heap.add(heapEntry) /*The heap will be
                sorted by score.*/
return top-k result
```

For the expanded/relaxed query $q(\lambda, \tilde{K}, W)$, where $q.\lambda$ is the location information of the query, $q.\tilde{K}$ is the set of query keywords (which is expanded/relaxed from $q.K$), $q.W$ is the set of weights on numerical attributes specified by the user. The function of Algorithm 2 is to efficiently search the top-k results that most satisfy user's needs and preferences. Algorithm 2 works as follows. First, the root node *root* is added to the maximum heap *heap*. Second, for each object

TABLE 4. The statics of the training set.

	Vocabulary	Avarage Length
Queries	7,175,000	4.37
Keywords	849,600	3.94

N in the *heap*, if N is a spatial object, that is, *root* is a leaf node, then N is added to *result* and the score of S_1, S_2 and integrated score *score* are calculated. Otherwise, N is an intermediate node. For each entity e in N, determine whether it contains the query keyword. If not, this branch is no longer traversed. Otherwise, S_2 is calculated using Equation (6) after obtaining the Skyline of its numerical attribute (Line 15). The value of S_1 and *score* are calculated by Equation (8) (Line 14) and Equation (7) (Line 16), respectively, then e is added to *heap* and iterates until the *heap* is empty or the size of *result* is greater than k. Lastly, the top-k result set is returned.

The time complexity of the Algorithm 2 is $O(kn)$, where $n = heap.size()$ and k is the number of results.

VI. EXPERIMENT

In this section, we systematically introduce the experiment settings and report the experimental results.

A. EXPERIMENT SETTING

We use two public Location-Based Social Networks (LBSNs) datasets, Yelp reviews dataset[1] and Foursquare dataset,[2] as the training data to train CGAN model. We reserve the text and user review information and extract the <query, keyword> pairs from these information. The size of <query, keyword> pairs is approximate to 8 million. We did token normalization (e.g., converting to lower-cased tokens, dropping special characters, etc.) and represented each query (and keyword) as a sequence of uni-gram terms, which yields a dictionary size of 7,175,000 for queries and 849,600 for keywords. The average length of queries and keywords is 4.37 and 3.94 tokens, respectively. We summarize the statistics of the training set in Table 4.

We then use the trained CGAN model to expand the user original query keywords on the following two datasets. The first is a real POI dataset captured from Yelp, a famous business review website in the United States, which contains business information, user evaluation, check-in time, and other information of restaurants, shopping centers, hotels, and other fields. These real POI data are processed into 174,567 POIs so that each POI has an ID, location information (in the form of longitude and latitude), text information, and numerical attributes. We took location information as spatial information, user comment information and POI category as text information, and randomly generated 5 random numbers between 0 and 1 as numerical attribute values. The second dataset comes from Foursquare. After data cleaning, the dataset contains 215,614 spatial objects. Each spatial object contains latitude and longitude information, keyword information such as steak, pizza, coffee, and four numerical attributes including price, environment, service, and rating. The characteristics of the test datasets are shown in Table 5.

[1]https://www.yelp.com/dataset
[2]https://foursquare.com/

TABLE 5. The characteristics of the test dataset.

Characteristics	Yelp	Foursquare
The total number of POIs	174,567	215,614
The number of all keywords	2,083,587	3,949,020
The number of all different keywords	57,437	84,716
The average number of keywords per POI	7	6

TABLE 6. Default values for parameters.

Parameter	Default value	Description
k	10	The number of top-k result objects
α	0.5	The adjustment parameter of location proximity and text relevance
β	0.7	The adjustment parameter of S_1 and S_2
$\|o.A\|$	4	The number of numerical attributes
$\|q.K\|$	4	The number of query keywords
$\|D\|$	10,000	The number of POIs

All experiments are implemented in Java. The computer is configured with 3.7GHZ CPU i7-8700k and Ubuntu 18.04.1 with 32GB RAM. The default values for the parameters are given in Table 6.

B. BASELINE COMPARISON METHODS

To give a comprehensive comparison, we implement two baseline spatial keyword query indexing algorithms, the one is IR-tree [17] and the other is IRS-tree [46] which can deal with the numerical attributes.

1) IR-TREE

It is an efficient hybrid index structure that can simultaneously handle both the textual and spatial information, which facilitates four major tasks in document searches, namely, spatial filtering, textual filtering, relevance computation and document ranking in a fully integrated manner. Besides, IR-tree allows searches to adopt different weights on textual and spatial relevance of documents at the run time and thus caters for a wide variety of applications. However, it just treats the numerical values as the textual keywords, which usually makes the query results unsatisfactory to the user's needs and preferences.

2) IRS-TREE

It is a hybrid index structure with Synopses' inverted R-tree, which can effectively handle a group of generic location-aware rank query (GLRQ) to return k objects satisfying the predicate ranked according to the ranking function and enable pruning of search space according to the satisfiability of the predicate. However, the query algorithm based on IRS-tree requires a precise range of numerical attributes, which may result in too few or no answer problem. In addition, users may not present an appropriate precise query range on numerical attributes.

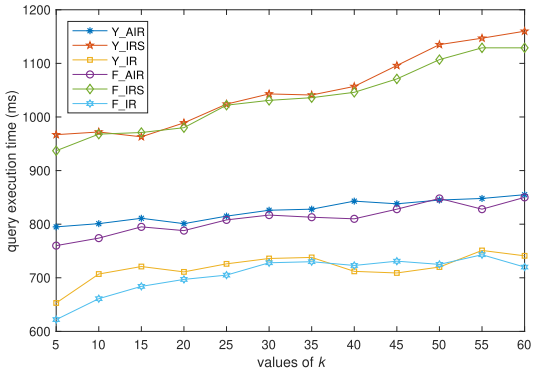

FIGURE 6. The impact of k value on query execution time.

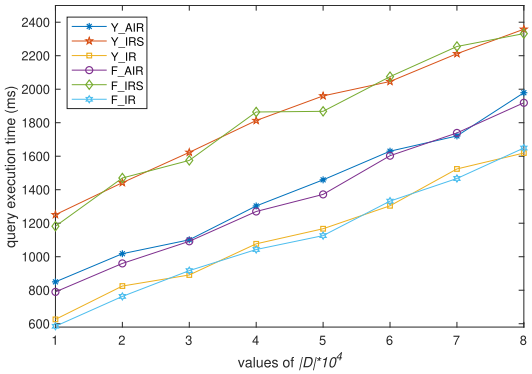

FIGURE 7. The impact of $|D|$ value on query execution time.

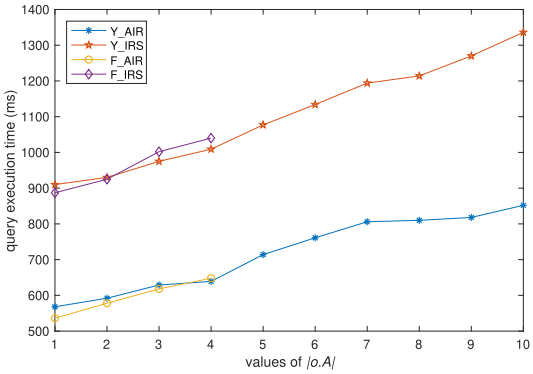

FIGURE 8. The impact of $|o.A|$ value on query execution time.

C. EXPERIMENTAL RESULTS AND ANALYSIS

This section mainly tests the efficiency and effectiveness of our proposed spatial keyword query model and corresponding hybrid index. We also compared our method with baseline methods described above over the same datasets.

1) THE EXPERIMENTS ON QUERY EFFICIENCY

Our main purpose in this group of experiments is to evaluate the impact of the number of query results k, the dataset size $|D|$, the number of numerical attributes $|o.A|$, and the number of query keywords $|q.K|$ on the query efficiency (i.e., query execution time).

a: THE IMPACT OF K ON QUERY EXECUTION TIME

The query execution time of our method and the two baselines on two datasets are evaluated by changing the number of query results from 5 to 60 with the interval of 5 and the experimental result is shown in Figure 6. Note that, in the following figures ''F/Y_index structure'' represents the experiment of each index on Foursquare/Yelp dataset, respectively.

It can be seen that the query efficiency of all methods becomes worth as the increase of the values of k. This is because the larger the value of k, the more candidate objects are indexed, so the longer the query execution time. It also can be seen that the query execution time of IR-tree is the shortest compared with the AIR-tree and IRS-tree, because it does not consider the numerical attributes and the semantic information, result in a great reduce of query execution time. AIR-tree is worse than IR-tree since it needs to calculate the user's satisfaction on numerical attributes. The query efficiency of AIR-tree is improved by 12.09% compared with IRS-tree index structure on the both two datasets.

b: THE IMPACT OF |D| ON QUERY EXECUTION TIME

This experiment aims to compare the query efficiency of different query algorithms by choosing the number of POIs from 10,000 to 80,000 with the interval of 10,000. The experimental results are shown in Figure 7.

From Figure 7, it can be observed that the query execution time increases dramatically as the size of dataset increases, which is due to the larger dataset, the more objects that need to be indexed, and thus it may take more time to process the numerical attributes. It also can be seen that AIR-tree has a much shorter query response time than IRS-tree because IRS-tree strictly restricts the precise range of numeric attributes, which leads to a rapid increase in query computation time while the Skyline method used in AIR-tree's numerical query can achieve fuzzy query to reduce query response time greatly.

c: THE IMPACT OF |o.A| ON QUERY EXECUTION TIME

This experiment aims to test the query efficiency of different query algorithms by varying the number of numerical attributes from 1 to 10.

The experimental result of Figure 8 shows that the query execution time increases with the number of numerical attributes increasing. This is because AIR-tree index structure requires to calculate the user's satisfaction on numerical attributes. Clearly, the more number of numerical attributes, the more time would be consumed. IRS-tree index structure is more time-consuming than AIR-tree for it considers the precise range of numerical attributes when dealing with the numerical attributes.

d: THE IMPACT OF |q.K| ON QUERY EXECUTION TIME

This experiment aims to test the impact of the number of query keywords on query execution time of different query algorithms by varying it from 1 to 8. The experimental results are shown in Figure 9.

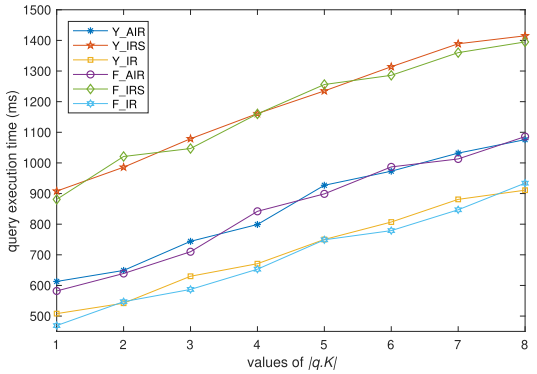

FIGURE 9. The impact of $|q.K|$ value on query execution time.

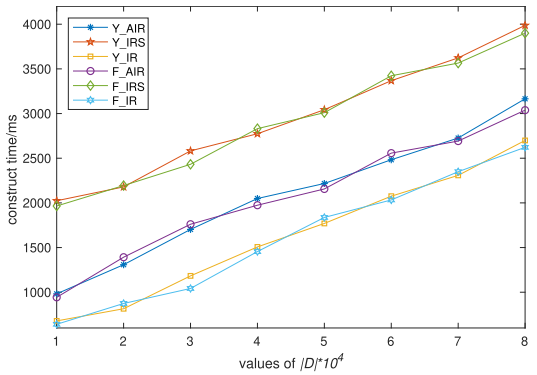

FIGURE 10. The impact of $|D|$ on the index building.

From Figure 9, it can be clearly seen that the query execution time increases in direct proportion to the number of query keywords. The reason is that no matter what kind of index structure, when the number of query keywords increases, more keywords that are semantically related to query keywords would be expanded into the collection of query keywords, so that the more objects containing query keywords need to be indexed, which result in the query response time increasing rapidly. IR-tree performs better than the others due to its simple index structure which does not need to consider processing much additional information (such as semantic information and numerical information). It should be noticed that the processing cost of IRS-tree index increases rapidly with the growth of $|q.K|$, since IRS-tree needs to scan more groups of objects containing the query keywords in the text document and falling into the precise query range of numerical attributes.

e: THE IMPACT OF |D| ON INDEX BUILDING

This experiment aims to test the impact of dataset size on the AIR-tree hybrid index building by varying the size of the dataset. The experimental results shown in Figure 10.

From Figure 10, it can be seen that the time cost for building the AIR-tree index is proportional to the size of dataset. IR-tree is the most efficient because it does not need to build *AttrFile* and *synopses* compared with AIR-tree and IRS-tree index structures, respectively. Despite taking the information of semantic information as well as numeric attribute information into account, AIR-tree is more efficient than IRS-tree in construction cost. The reason is that IRS-tree requires the synopses tree to be combined with other indexes to complete the query and it also needs to consider more numerical attributes' precise range when dealing with numerical attributes, which makes it take the longest time to be built.

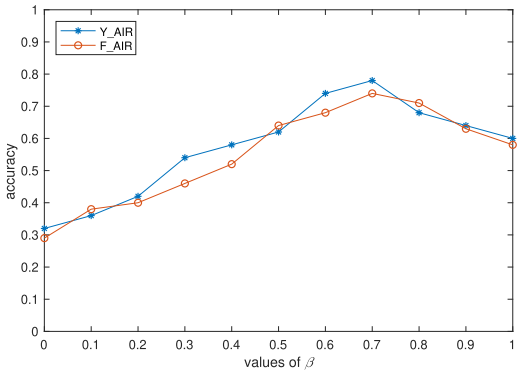

FIGURE 11. The impact of value β on accuracy.

2) THE EXPERIMENTS ON ACCURACY

The purpose of this group of experiments is to evaluate the impact of parameters β and k on accuracy of different query algorithms.

Since AIR-tree is a high-dimensional approximate query index, some semantically related results without exactly text matching would be obtained for a given query q. Therefore, we need to evaluate the accuracy of the query results. We use the users satisfaction to measure the accuracy of different query algorithms: First, we randomly picked 10 spatial objects from the dataset as the test queries. And then, for each query, the IR-tree, AIR-tree, and IRS-Tree indexes are used to retrieve the top-10 most relevant objects, respectively. In this way, each query q_i corresponds to a target set H_i of 30 objects which likely to contain a good mix of relevant and irrelevant objects to q_i (if there are duplicates, remove the duplicates and add new objects randomly). Next, for each q_i, we asked 10 teachers, 30 graduate students, and 60 undergraduates to identify the top-10 objects from H_i that they thought were most relevant to q_i. Here, $I(q_i)$ represents the top-10 objects labeled by users for query q_i as the ground truth, and $R(q_i)$ refers to the top-10 objects retrieved by IR-tree, AIR-tree, and IRS-tree, respectively. Thus, the accuracy can be defined as follows:

$$accuracy = \frac{|I(q_i)| \bigcap R(q_i)|}{10} \tag{9}$$

a: THE IMPACT OF β ON ACCURACY

Since Equation (7) is only used by our AIR-tree based query evaluation, we just test the impact of parameter β on the accuracy for our method over the two datasets. Figure 11 shows the accuracy of our method for different values of β.

It can be seen that the accuracy reaches the peak at $\beta = 0.7$ for both Yelp and Foursquare datasets and the corresponding

TABLE 7. The accuracy of top-10 spatial object results.

	Yelp	Foursquare
AIR-tree	0.78	0.74
IR-tree	0.52	0.53
IRS-tree	0.64	0.62

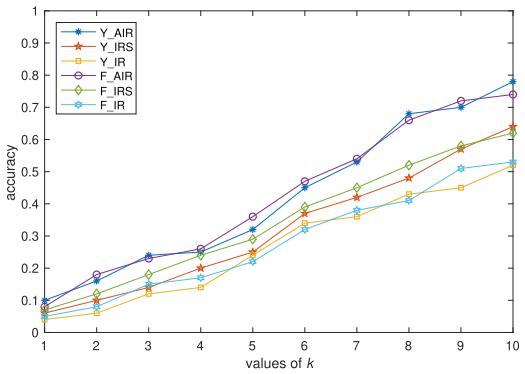

FIGURE 12. The effect of k value on accuracy.

accuracy are 0.78 and 0.74 respectively, which demonstrates that our method considers the user's satisfaction on numerical attributes is very helpful for improving the accuracy. Furthermore, we can also observe that the accuracy corresponding to $\beta = 1$ (i.e., the query algorithm only consider the location proximity and text semantic similarity when evaluating the query results) is better than that corresponding to $\beta = 0$ (i.e., the query algorithm only consider the user's satisfaction on numerical attributes when evaluating the query results), which indicates that the combination of location proximity and text semantic similarity is more important than the user's satisfaction on numerical attributes for evaluating the query results and this is reasonable in reality.

b: THE IMPACT OF k ON ACCURACY

The performance of the AIR-tree against the baselines is reported in terms of accuracy for top-10 objects in Table 7. We also test the accuracy of the top-k ($k = 1, 2, \ldots, 10$) spatial objects in Figure 12 (the parameter β is fixed to be 0.7).

As shown in Figure 12, AIR-tree index structure proposed in this paper outperforms all baselines on the two datasets. When k ranged from 1 to 10, the average accuracy of AIR, IRS and IR is 0.4225, 0.3345 and 0.2760, respectively, so AIR-tree improves the averaged accuracy by 14.65% and 8.80% compared with IR-tree and IRS-tree, respectively. The significant improvement of AIR-tree indicates that it can well satisfy the user's needs in personality and semantic for the top-k results. This is because we integrally consider the location proximity, text semantic similarity, and user's satisfaction on numerical attributes and integrate these aspects to build a hybrid index structure. Moreover, AIR-tree not only reduces the burden on the user to specify the exact query range of numerical attributes, but also improves the performance compared with IRS-tree. The result of IR-tree is the worst on accuracy because it does not consider the semantic approximation and user's satisfaction on numerical attributes. Although IRS-tree can process the numerical attribute values, it does not consider the semantic relevancy of query results and user's preferences on numerical attributes. It also can be seen that the accuracy of each algorithm gradually improves with the increases of k. This is because some relevant objects that not ranked in front of the result list would not be provided by the algorithm when k is small, while they would be in the result set when k is large, so that the overlap between the set of objects obtained by the algorithm and the set of objects marked by the user would grow as k increases.

VII. CONCLUSION

This paper proposes a spatial keyword querying approach by using CGAN and Skyline to realize the approximation and personalization of query results. The experimental results on real datasets reveal that the proposed algorithm not only supports the exact matching of spatial keywords but also supports the semantic approximation query, and can deal with the numerical attributes in the text information as well, which can satisfy the user's needs and preferences more closely.

In future work, CGAN will be further optimized in the pre-training stage by incorporating more context information such as social network into the model to obtain a better ability to expand query keywords. The concept of diversification will be introduced and studied to make the query results satisfy the user's query requirements and preferences more efficiently. The weights between spatial location proximity and text similarity will also be inferred based on user's preferences.

REFERENCES

[1] 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.
[2] C. Zhang, Y. Zhang, W. Zhang, and X. Lin, "Inverted linear quadtree: Efficient top k spatial keyword search," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1706–1721, Jul. 2016.
[3] K. Zheng, H. Su, B. Zheng, S. Shang, J. Xu, J. Liu, and X. Zhou, "Interactive Top-k spatial keyword queries," in *Proc. IEEE 31st Int. Conf. Data Eng.*, Seoul, South Korea, Apr. 2015, pp. 423–434.
[4] G. Cong and C. S. Jensen, "Querying geo-textual data: Spatial keyword queries and beyond," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, San Francisco, CA, USA, Jun./Jul. 2016, pp. 2207–2212.
[5] I. De Felipe, V. Hristidis, and N. Rishe, "Keyword search on spatial databases," in *Proc. IEEE 24th Int. Conf. Data Eng.*, Cancún, Mexico, Apr. 2008, pp. 656–665.
[6] Y. Lu, J. Lu, G. Cong, W. Wu, and C. Shahabi, "Efficient algorithms and cost models for reverse Spatial-Keywordk-Nearest neighbor search," *ACM Trans. Database Syst.*, vol. 39, no. 2, pp. 1–46, May 2014.
[7] G. Li, J. Xu, and J. Feng, "Keyword-based k-nearest neighbor search in spatial databases," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, Maui, HI, USA, Oct./Nov. 2012, pp. 2144–2148.
[8] X. Wang, Y. Zhang, W. Zhang, X. Lin, and Z. Huang, "Skype: Top-K spatial-keyword publish/subscribe over sliding window," *Proc. VLDB Endowment*, vol. 9, no. 7, pp. 588–599, Mar. 2016.
[9] D. Zhang, Y. M. Chee, A. Mondal, A. K. H. Tung, and M. Kitsuregawa, "Keyword search in spatial databases: Towards searching by document," in *Proc. IEEE 25th Int. Conf. Data Eng. (ICDE)*, Shanghai, China, Mar./Apr. 2009, pp. 688–699.
[10] G. Cong, C. S. Jensen, and D. Wu, "Efficient retrieval of the top-k most relevant spatial Web objects," *Proc. VLDB Endowment*, vol. 2, no. 1, pp. 337–348, Aug. 2009.
[11] L. Chen, X. Lin, H. Hu, C. S. Jensen, and J. Xu, "Answering why-not questions on spatial keyword top-k queries," in *Proc. IEEE 31st Int. Conf. Data Eng.*, Seoul, South Korea, Apr. 2015, pp. 279–290.

[12] H.-Y. Kwon, H. Wang, and K.-Y. Whang, "G-index model: A generic model of index schemes for top-k spatial-keyword queries," *World Wide Web*, vol. 18, no. 4, pp. 969–995, Jun. 2014.
[13] H. K.-H. Chan, C. Long, and R. C.-W. Wong, "On generalizing collective spatial keyword queries (Extended Abstract)," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Macao, China, Apr. 2019, pp. 2115–2116.
[14] Y. Gao, J. Zhao, B. Zheng, and G. Chen, "Efficient collective spatial keyword query processing on road networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 469–480, Feb. 2016.
[15] S. Park and S. Park, "Reverse collective spatial keyword query processing on road networks with G-tree index structure," *Inf. Syst.*, vol. 84, pp. 49–62, Sep. 2019.
[16] Y. Wu, J. Xu, L. Tu, M. Luo, Z. Chen, and N. Zheng, "Reverse collective spatial keyword querying (short paper)," in *Proc. 14th EAI Int. Conf. Collaborative Comput., Netw., Appl. Worksharing (CollaborateCom)*, Shanghai, China, Dec. 2018, pp. 211–221.
[17] Z. Li, K. C. K. Lee, B. Zheng, W.-C. Lee, D. Lee, and X. Wang, "IR-tree: An efficient index for geographic document search," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 4, pp. 585–599, Apr. 2011.
[18] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An efficient and robust access method for points and rectangles," in *Proc. Int. Conf. Manage. Data (ACM SIGMOD)*, Atlantic City, NJ, USA, May 1990, pp. 322–331.
[19] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. Annu. Meeting (SIGMOD)*, Boston, MA, USA, Jun. 1984, pp. 47–57.
[20] F. Li, B. Yao, M. Tang, and M. Hadjieleftheriou, "Spatial approximate string search," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1394–1409, Jun. 2013.
[21] J. B. Rocha, Jr., O. Gkorgkas, S. Jonassen, and K. Nørvåg, "Efficient processing of top-k spatial keyword queries," in *Proc. 12th Int. Symp. Adv. Spatial Temporal Databases (SSTD)*, Minneapolis, MN, USA, Aug. 2011, pp. 205–222.
[22] B. Yao, F. Li, M. Hadjieleftheriou, and K. Hou, "Approximate string search in spatial databases," in *Proc. IEEE 26th Int. Conf. Data Eng. (ICDE)*, Long Beach, CA, USA, Mar. 2010, pp. 545–556.
[23] S. Ray and B. G. Nickerson, "Dynamically ranked top-k spatial keyword search," in *Proc. 3rd Int. ACM SIGMOD Workshop Manag. Mining Enriched Geo-Spatial Data (GeoRich)*, San Francisco, CA, USA, Jun./Jul. 2016, pp. 6:1–6:6.
[24] Z. Qian, J. Xu, K. Zheng, P. Zhao, and X. Zhou, "Semantic-aware top-k spatial keyword queries," *World Wide Web*, vol. 21, no. 3, pp. 573–594, Jun. 2017.
[25] X. Zang, P. Hao, X. Gao, B. Yao, and G. Chen, "Qdr-tree: An efficient index scheme for complex spatial keyword query," in *Proc. 29th Int. Conf. Database Expert Syst. Appl. (DEXA)*, Regensburg, Germany, Sep. 2018, pp. 390–404.
[26] H. Kim, J. Kim, and P. Lim, "Towards perfect text classification with wikipedia-based semantic naïve Bayes learning," *Neurocomputing*, vol. 315, pp. 128–134, Nov. 2018.
[27] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," in *Proc. Adv. Neural Inf. Process. Syst., Neural Inf. Process. Syst., Natural Synth. (NIPS)*, Vancouver, BC, Canada, Dec. 2001, pp. 601–608.
[28] D. M. Blei and J. D. Lafferty, "Dynamic topic models," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*, Pittsburgh, PA, USA, Jun. 2006, pp. 113–120.
[29] S. Kim and P. Smyth, "Hierarchical Dirichlet processes with random effects," in *Proc. 20th Annu. Conf. Neural Inf. Process., Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, vol. 19, Dec. 2006, pp. 697–704.
[30] L. Du, W. L. Buntine, and H. Jin, "Sequential latent Dirichlet allocation: Discover underlying topic structures within a document," in *Proc. IEEE Int. Conf. Data Mining*, Sydney, NSW, Australia, Dec. 2010, pp. 148–157.
[31] B. Hu, M. Jamali, and M. Ester, "Spatio-temporal topic modeling in mobile social media for location recommendation," in *Proc. IEEE 13th Int. Conf. Data Mining*, Dallas, TX, USA, Dec. 2013, pp. 1073–1078.
[32] Q. Liu, Y. Ge, Z. Li, E. Chen, and H. Xiong, "Personalized travel package recommendation," in *Proc. IEEE 11th Int. Conf. Data Mining*, Vancouver, BC, Canada, Dec. 2011, pp. 407–416.
[33] L. Jing, M. K. Ng, and J. Z. Huang, "Knowledge-based vector space model for text clustering," *Knowl. Inf. Syst.*, vol. 25, no. 1, pp. 35–55, Oct. 2009.
[34] W. Hua, Z. Wang, H. Wang, K. Zheng, and X. Zhou, "Short text understanding through lexical-semantic analysis," in *Proc. IEEE 31st Int. Conf. Data Eng.*, Seoul, South Korea, Apr. 2015, pp. 495–506.
[35] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. 1st Int. Conf. Learn. Representations (ICLR)*, Scottsdale, AZ, USA, May 2013, pp. 1–12.
[36] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 27th Annu. Conf. Neural Inf. Process. Syst., Adv. Neural Inf. Process. Syst*, Lake Tahoe, NV, USA, Dec. 2013, pp. 3111–3119.
[37] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Doha, Qatar, 2014, pp. 1532–1543.
[38] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*. [Online]. Available: https://arxiv.org/abs/1411.1784
[39] M.-C. Lee, B. Gao, and R. Zhang, "Rare query expansion through generative adversarial networks in search advertising," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, London, U.K., Aug. 2018, pp. 500–508.
[40] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Annu. Conf. Neural Inf. Process. Syst., Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, Dec. 2014, pp. 2672–2680.
[41] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Denver, CO, USA, vol. 12, Nov./Dec. 1999, pp. 1057–1063.
[42] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," 2014, *arXiv:1409.1259*. [Online]. Available: https://arxiv.org/abs/1409.1259
[43] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proc. 17th Int. Conf. Data Eng.*, Heidelberg, Germany, Apr. 2001, pp. 421–430.
[44] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Proc. Annu. Conf. Neural Inf. Process. Syst., Adv. Neural Inf. Process. Syst.*, Barcelona, Spain, Dec. 2016, pp. 2226–2234.
[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–15.
[46] X. Liu, L. Chen, and C. Wan, "LINQ: A framework for location-aware indexing and query processing," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1288–1300, May 2015.

XIANGFU MENG was born in 1981. He received the Ph.D. degree from Northeastern University, China, in 2010. He is currently a Full Professor and a Ph.D. Supervisor with Liaoning Technical University, China. His research interests include spatial data management, recommender systems, and big data visualization.

PAN LI was born in 1996. She is currently pursuing the master's degree with Liaoning Technical University, China. Her research interests include spatial keyword query and spatial recommendation systems.

XIAOYAN ZHANG was born in 1983. She received the M.S. degree from Northeastern University, China. She is currently pursuing the Ph.D. degree with Liaoning Technical University, China. Her research interests include spatial data analysis, city computing, and deep learning.