Re-CoSKQ: Towards POIs Recommendation Using Collective Spatial Keyword Queries

Ramon Hermoso[*]
rhermoso@unizar.es
Department of Computer Science and Systems Engineering, University of Zaragoza
Zaragoza, Spain

Sergio Ilarri
silarri@unizar.es
Department of Computer Science and Systems Engineering, I3A, University of Zaragoza
Zaragoza, Spain

Raquel Trillo-Lado
raqueltl@unizar.es
Department of Computer Science and Systems Engineering, I3A, University of Zaragoza
Zaragoza, Spain

ABSTRACT

The goal of collective spatial keyword queries is to retrieve, from a spatial database, a group of spatial items such that the description of the items included in that set (typically based on the use of keywords) is completely covered by the query's keywords. Moreover, it ensures that the items retrieved are as near as possible to the query location and have the lowest inter-item distances. We argue that using this concept in the field of recommender systems could be useful. Therefore, in this position paper, we outline the idea of Re-CoSKQ, an adaptation of Collective Spatial Keyword Query (CoSKQ) for recommender systems in the tourism domain to provide the user with a set of Points of Interest (POIs) that satisfy his/her queries both geographically and semantically.

CCS CONCEPTS

• **Retrieval tasks and goals** → **Recommender systems**.

KEYWORDS

Collective spatial keyword querying, recommender systems, tourism

1 INTRODUCTION

Recommender systems (RS) have been studied for several decades, aiming to facilitate item selection as part of the user's decision-making processes [11]. One of the hard challenges of recommender systems is to provide successful responses to user queries, especially when little information is available. In most RS approaches, algebraic operations with user-item rating matrices allow predicting the future likeness of new items for a user (e.g., using collaborative filtering, content-based, or hybrid approaches). However, when the suitability of the suggested items depends on different features such as the location of items and users, textual descriptions of items, or the (sometimes blurry) query description, those approaches face new problems to address. For example, for the recommendation of points of interest (POIs), the location of the items and the user, as well as other context attributes, may play a key role [6].

The idea of Collective Spatial Keyword Querying (CoSKQ) emerged some years ago as a promising technique to query spatial databases containing information about items and their location [2]. It puts forward a smart solution to retrieve a group of spatial items such that the description of the items included in that set (typically based on the use of keywords) is completely covered by the query's keywords and assures that the items are as near as possible to the query location and have the lowest inter-item distances.

[*]All authors contributed equally to this research.

We believe that exploiting spatial keyword querying as a basis to build recommender systems is an interesting research avenue to explore. Therefore, combining both fields of research, in this position paper we present the idea of Re-CoSKQ, a recommender system that uses CoSKQ to provide a set of items that semantically covers the keywords of a query (even if they do not match perfectly) and minimizes the cost, in terms of the distance to get to them and the similarity between query keywords and item descriptions.

As a problem statement, let us consider a set $U = \{u_1, ..., u_n\}$ of users spending their time in a city as tourists. Let $\mathbb{O} = \{o_1, ..., o_m\}$ be a set of POIs, i.e., spots with some kind of relevant attraction for visitors. Examples of POIs could be museums, monuments, parks, or buildings with some historical flavour, just to mention a few. Now, let $o_i.\kappa = \{k_1, ..., k_j\}$ be a set of keywords with which a POI $o_i \in \mathbb{O}$ is described. These keywords can usually be retrieved in an automated way by using semantically-annotated resources. Moreover, every POI $o_i \in \mathbb{O}$ is placed in a location denoted by $o_i.\lambda$.

Re-CoSKQ uses collective spatial keyword querying in order to cope with the location of POIs and users and also with the similarity between the keywords in the user's query and the description of the POIs. Let $q = \langle \lambda, \kappa \rangle$ be a user's query, where $q.\lambda$ represents the user's location and $q.\kappa$ stands for the query split in keywords (only relevant words for the search are taken into account). The main goal is to provide a method to return a set of items $\mathbb{O}' \subseteq \mathbb{O}$ which semantically covers the keywords in $q.\kappa$ and also ensures that their cost, in terms of distance –between the POIs and the user who issued the query– and the similarity of terms, is minimal.

The next sections intend to shed some light into the problem and present the approach we have envisioned to deal with it. Specifically, the rest of the paper is structured as follows. First, Section 2 we revise the concept of CoSKQ. Then, in Section 3, we present the Re-CoSKQ approach. In Section 4, we sketch an evaluation proposal. Finally, in Section 5, we conclude with a summary and some future work.

2 BACKGROUND: CoSKQ

As we have previously stated, CoSKQ attempts to find the solution to the problem of retrieving a group of spatial objects that collectively match the user preferences given specific locations (of the user and also of the objects) and a set of keywords. The method is designed to work with spatial databases, so it does much effort on providing an efficient computation, in terms of the data structure used and how data are accessed [2, 3]. Although going in depth on the subtle considerations of the method is out of the scope of this paper, we summarize how it works applied to our domain.

It uses the concept of IR-tree data structures [4] to efficiently store information about POIs. This type of structure allows indexing objects and the keywords which describe them as well as their spatial position. IR-trees are a type of balanced trees in which each leaf node contains an item o (a POI object), a bounding rectangle of o and an item identifier, while each non-leaf node in the tree contains a pointer to a child node, a *Minimum Bounding Rectangle* (MBR) of all rectangles in entries of the child node, and an item identifier containing the set of all keywords in the entries of the child node. Moreover, each leaf node contains a pointer to an inverted file with the keywords that describe the POIs stored in that node. Figure 1 depicts an example for which CoSKQ may offer a solution with a query q and a set of POIs $\{o_1, ..., o_{10}\}$. Figures 2 and 3 show how the data are geographically partitioned and stored in an IR-tree.

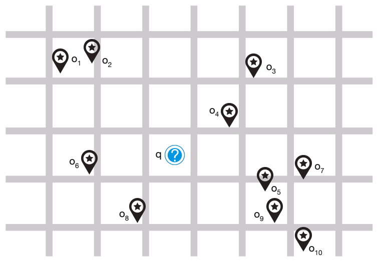

Figure 1: Example of a possible scenario

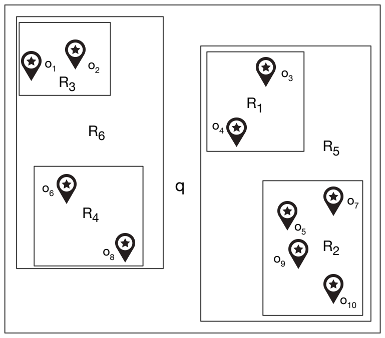

Figure 2: Item positioning for the example

CoSKQ presents different algorithmic solutions based on minimizing a cost function. The chosen cost function may vary depending on the authors of each specific proposal and the scenario where it is applied. Different cost functions, taking into account the distances between items and query locations, can be found in [2, 3]. It has been proved that solving a spatial group keyword query is an NP-complete problem [2], i.e., the performance of an exact algorithm does not present itself as a reasonable solution, in terms of running time and I/O cost [7]. For that reason, some approximation algorithms have been developed to calculate the output sets of objects [2, 3, 7, 12]. Besides, in special cases, the application of an exact algorithm may be plausible, especially when the number of keywords in the query is small. Some exact algorithms, based on dynamic programming for minimizing the cost function are presented in [2, 3, 7].

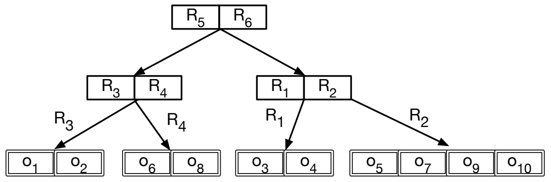

Figure 3: Resulting IR-tree containing data for the example

3 Re-CoSKQ APPROACH

We present Re-CoSKQ as an instantiation of the CoSKQ problem, especially designed for recommendations in the tourism domain (i.e., the user is a tourist and the items are points of interest that the user may want to visit). The most common instantiation of CoSKQ assumes that the set of keywords describing the POIs in the query result must contain, at least, all the keywords contained in the query [2]. Formally, $q.\kappa \subseteq \cup_{o'_i \in \mathbb{O}'} o'_i.\kappa$, where $\mathbb{O}'$ is the set of POIs calculated as a result of a user issuing the query q; for simplicity, from now on, we will use $o \in \mathbb{O}'$ to avoid o'_i. However, there are scenarios in which this assumption must hold some more hard constraints. For instance, when tackling a recommendation problem, we need to ensure not only that the keyword query is covered by the resulting $\mathbb{O}'$ but also that both the maximum distance between the query location and any of the POIs in $\mathbb{O}'$ and the maximum distance between any two POIs in $\mathbb{O}'$ are minimized.

Moreover, in this paper, we do not assume that $q.\kappa$ can be fully covered. Actually, we claim that this assumption may derive in empty sets in many recommendation scenarios where queries are expressed, for instance, with different vocabularies, or where they cannot be easily solved with the given descriptions of POIs. Thus, we believe that it is important to provide query outcomes even when full keyword coverage is not possible. In order to do that, we propose to use a similarity function to calculate how similar the keywords in $q.\kappa$ are compared to those in $\cup_{o \in \mathbb{O}'}$. For example, given $q.\kappa = \{outdoors, animals, kids\}$, if located nearby, one of the POIs included in the outcome could be a zoo, which could be described by a set of keywords $\{open\text{-}air, birds, snakes, mammals, family\}$. This object would never be returned using a classic CoSKQ approach, but considering the semantic similarity between keywords one can easily observe that the terms are related, since *birds*, *snakes* and *mammals* are types of *animals*, *outdoors* and *open-air* are synonyms and *kids* are part of *families*. We will present how to cope with this when presenting different cost functions.

3.1 Cost Analysis

Re-CoSKQ attempts to minimize the cost of finding an appropriate set of POIs for a given query q. This cost is modelled as a function that depends on distances between the query and the locations of POIs as well as between the keywords. Different equations have been proposed to model cost in the CoSKQ problem [1, 2, 10]. In the following, we redefine some of them for the Re-CoSKQ problem.

TYPE 1. A linear combination of the maximum distance between the query location and any POI in $\mathbb{O}'$, the maximum pairwise distance between any two POIs in $\mathbb{O}'$, and the maximum of the semantic distance between the query keywords ($q.\kappa$) and the set of keywords in $\mathbb{O}'$, i.e. $\cup_{o \in \mathbb{O}'} o.\kappa$. It is formally defined in Eq. 1.

$$\begin{aligned} cost(q, \mathbb{O}') = \alpha \cdot \max_{o \in \mathbb{O}'} [dist(q.\lambda, o.\lambda)] + \beta \cdot \max_{o_1, o_2 \in \mathbb{O}'} [dist(o_1, o_2)] \\ + \omega \cdot \max_{k_1 \in q.\kappa, k_2 \in \cup_{o \in \mathbb{O}'} o.\kappa} [dist(k_1, k_2)] \end{aligned} \tag{1}$$

where $\alpha + \beta + \omega = 1$ are weights to denote the relevance of each of the three types of distances involved, which allow adding up distances which may have different ranges of values.

TYPE 2. This type of function defines cost as the maximum of the three factors in the TYPE 1 function; i.e., the highest value between the maximum distance among the query location and any POI in $\mathbb{O}'$, the maximum pairwise distance between any two POIs in $\mathbb{O}'$, and the maximum of the semantic distance between query keywords ($q.\kappa$) and the set of keywords in $\cup_{o \in \mathbb{O}'} o.\kappa$. This is formally defined by Eq. 2, where again weights α, β and ω are used.

$$\begin{aligned} cost(q, \mathbb{O}') = \max \Big\{ \alpha \cdot \max_{o \in \mathbb{O}'} [dist(q.\lambda, o.\lambda)], \beta \cdot \max_{o_1, o_2 \in \mathbb{O}'} [dist(o_1, o_2)], \\ \omega \cdot \max_{k_1 \in q.\kappa, k_2 \in \cup_{o \in \mathbb{O}'} o.\kappa} [dist(k_1, k_2)] \Big\} \end{aligned} \tag{2}$$

TYPE 3. This function uses a min-max approach, linearly combining the minimum distance between the query location and any POI with the maximum values for pairwise distance between any two POIs in $\mathbb{O}'$ and the semantic distance between query keywords ($q.\kappa$) and the set of keywords in $\mathbb{O}'$, i.e., $\cup_{o \in \mathbb{O}'} o.\kappa$ (see Eq. 3).

$$\begin{aligned} cost(q, \mathbb{O}') = \alpha \cdot \min_{o \in \mathbb{O}'} [dist(q.\lambda, o.\lambda)] + \beta \cdot \max_{o_1, o_2 \in \mathbb{O}'} [dist(o_1, o_2)] \\ + \omega \cdot \max_{k_1 \in q.\kappa, k_2 \in \cup_{o \in \mathbb{O}'} o.\kappa} [dist(k_1, k_2)] \end{aligned} \tag{3}$$

again with $\alpha + \beta + \omega = 1$.

TYPE 4. This is a unified cost function, adapted from [3], that generalizes types 1 to 3 in one function. It is presented in Eq. 4.

$$\begin{aligned} cost(q, \mathbb{O}') = \Bigg[\Bigg(\alpha \cdot \Big(\sum_{o \in \mathbb{O}'} (dist(q.\lambda, o.\lambda))^{\phi_1} \Big)^{\frac{1}{\phi_1}} \Bigg)^{\phi_2} \\ + \Bigg(\beta \cdot \max_{o_1, o_2 \in \mathbb{O}'} dist(o_1, o_2) \Bigg)^{\phi_2} \\ + \Bigg(\omega \cdot \max_{k_1 \in q.\kappa, k_2 \in \cup_{o \in \mathbb{O}'} o.\kappa} dist(k_1, k_2) \Bigg)^{\phi_2} \Bigg]^{\frac{1}{\phi_2}} \end{aligned} \tag{4}$$

with $\alpha + \beta + \omega = 1$, $\phi_1 \in \{-\infty, 1, \infty\}$ and $\phi_2 \in \{1, \infty\}$. The ϕ_1 and ϕ_2 values stand for tuning parameters, allowing to describe the previous cost functions (types 1-3) by varying their values. For example, an instantiation with $\alpha, \beta, \omega = \frac{1}{3}$, $\phi_1 = \infty$ and $\phi_2 = 1$ results in a Type 1 cost function with the weights α, β, and ω indicated:

$$\begin{aligned} cost(q, \mathbb{O}') = \frac{1}{3} \Bigg(\max \sum_{o \in \mathbb{O}'} (dist(q.\lambda, o.\lambda)) + \\ + \max_{o_1, o_2 \in \mathbb{O}'} dist(o_1, o_2) + \max_{k_1 \in q.\kappa, k_2 \in \cup_{o \in \mathbb{O}'} o.\kappa} dist(k_1, k_2) \Bigg) \end{aligned}$$

3.2 Distance Analysis

As we have pointed out, there exist different distance functions needed to calculate the cost in Re-CoSKQ. Analyzing any of the proposed cost functions, we can observe that there are three different distance instantiations, as we explain in the following.

Location distance. ($dist(q.\lambda, o.\lambda)$) refers to the physical distance between the query location and a POI's location. It can be calculated with different geometrical approaches. In the following, we point out some possible functions.

Euclidean distance. It is probably the most common distance function used in the literature for many different types of problems and domains. It is formally defined by Eq. 5:

$$dist(q.\lambda, o.\lambda) = \sqrt{\sum_{i=1}^{n} (q.\lambda_i - o.\lambda_i)^2} \tag{5}$$

We assume that the position of queries and POIs are given by a pair of coordinates $\langle lat, long \rangle$. This distance may work well when the routes between POIs are roughly calculated or the users can walk straight from any location to another.

$\mathcal{L}_1$-Norm. It is another well-known distance function, also known as *Manhattan distance*. It calculates the sum of the magnitudes of the vectors in a space, i.e., the sum of absolute difference of the components of the vectors (see Eq. 6).

$$dist(q.\lambda, o.\lambda) = \sum_{i=1}^{n} |q.\lambda_i - o.\lambda_i| \tag{6}$$

We use 2-dimension spaces, denoted by location coordinates. This distance may be suitable for grid-based scenarios, e.g., POIs in a city connected by roads/paths, or halls in a museum linked by corridors.

Geodesic distance. It is the type of function we need if we use a graph to model how POIs and users are connected. The geodesic distance is defined as the shortest path between two vertices in a graph. This is useful when modeling a scenario with weighted edges, since some extra information can be added (e.g., about congested routes or crowded halls). Many algorithms can be used to calculate shortest paths in graphs (e.g., the Dijkstra's algorithm).

POI-to-POI distance. ($dist(o_1, o_2)$) could also be called intra-POI distance, since it calculates the distance between two POIs. Note again that the location of $o \in \mathbb{O}'$ is denoted by $o.\lambda$. As we assume a 2-dimension space in Re-CoSKQ, we can reduce the calculation of this distance to the problem of calculating the location distance. Thus, the same functions described above may apply to this case.

Term distance. ($dist(k_1, k_2)$) is the distance we use to calculate how similar two different keywords are. In this case, we compare the query keywords ($q.\kappa$) and the keywords in $\cup_{o \in \mathbb{O}'} o.\kappa$. In the cost function, we try to minimize the maximum distance between the $q.\kappa$ set and $o.\kappa$ in a pairwise basis. In order to calculate the similarity between keywords, we adhere to ontology-based measures, typically used in semantic web approaches. This type of measures usually calculates the similarity according to structured knowledge defined by an ontology. In the following, we propose some functions that we consider to be suitable for the Re-CoSKQ problem; once the similarity has been estimated, we should provide a way to calculate the distance associated to it, such as $dist(k_1, k_2) = 1 - sim(k_1, k_2)$.

Similarity based on concept closeness. This measure takes into account the closeness of the concepts in the hierarchical tree representing the ontology. It is based on the *relatedness* property presented in [8] and is defined as $sim(k_1, k2) = 1 - \frac{sp(k_1, k_2)}{2D}$, where $sp(\cdot)$ is a function that returns the shortest path between the two terms in the ontology tree and $2D$ denotes the maximum distance between any two concepts in the ontology (D is the ontology depth).

Similarity based on closeness and concept depth. This measure, proposed in [9], takes into account the closeness of keywords in the ontology but also the depth in the ontology tree where they can be found (see Eq. 7). It assumes that the semantics of concepts are more general in higher levels. Thus, the higher we find the concept in the ontology the lower the similarity while, on the contrary, the lower we find the concepts the higher the similarity.

$$sim(k_1, k_2) = \begin{cases} e^{-\alpha l} \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & if \quad k_1 \neq k_2 \\ 1 & otherwise \end{cases} \tag{7}$$

where l is the shortest path between k_1 and k_2 and h is the depth of the least common subsumer of both concepts. Parameters $\alpha, \beta > 0$ are weights to modulate the contribution of these factors.

3.3 Outline of Processing Issues for Re-CoSKQ

Several algorithms address the problem of implementing CoSKQ. CoSKQ is an NP-complete problem, so exact algorithms (e.g., the linear programming approaches presented in [2, 3]) only make sense when the number of query keywords is low. However, on average conditions, an approximate algorithm is needed. Different approaches using greedy techniques and pruning steps have been presented to reduce the needed resources. Due to lack of space, we omit further details and refer the reader to [2, 3] for further revision on approximate algorithms for CoSKQ. Re-CoSKQ needs to tackle an optimization problem to try to minimize the cost function.

4 EVALUATION PROPOSAL

Most works on CoSKQ focus their evaluation on measuring the performance (in terms of running time) and approximation ratio. Nevertheless, when applying this approach to recommender systems we are not only interested in these issues, as the quality of the recommendation is also key. An interesting problem is that full coverage is assumed in classic CoSKQ; that is, $\cup_{o \in \mathbb{O}'} o.\kappa$ is assumed to contain, at least, all keywords in $q.\kappa$. This is not the case of Re-CoSKQ, where the coverage is estimated by keyword similarity. Moreover, the evaluation usually needs a ground truth to compare with, in order to be able to calculate accuracy metrics such as precision and recall. As far as we know, there is no dataset annotated with this type of information. Thus, we propose to first define a set of interesting and representative keyword queries and then manually annotate a dataset containing POIs descriptions with the keywords by assigning each POI to a set of predefined categories (much smaller than the number of keywords) defined based on the queries that have been selected for evaluation, in order to define a dataset with information that can represent a suitable ground truth to compare with. Precision and recall may be calculated by comparing the retrieved POIs according to the categories specified by the user in the query. Regarding the items, there are many datasets that contain information about geographic locations and keyword descriptions; a tailored synthetic dataset could also be generated by using DataGenCARS [5]. All this could be complemented with a user-centered evaluation.

The main idea in the empirical evaluation is to show the benefits of the proposal and test how different cost functions behave, tuning different parameters. We are also interested in the scalability of the proposal, so tests with different numbers of query keywords and POIs (as well as simultaneous queries/users) should be carried out. Moreover, in order to check the feasibility concerning the use of resources, we will use exact and approximate algorithms to test their performance (running time and approximation ratio).

5 CONCLUSIONS AND FUTURE WORK

In this position paper, we have presented the idea of Re-CoSKQ, a collective spatial keyword query approach for recommender systems, where keyword coverage is not assumed, by considering keyword similarity. We have tackled the problem as a minimization problem, for which we have defined some cost functions.

We are currently working on an empirical evaluation to test the approach and its benefits over other POI recommendation approaches. Furthermore, we would like to extend the approach to group recommendation; that is, different users in different locations will issue their queries and the opportunity of group visits (groups of people visiting the same items together) will be explored, so the problem becomes more complex, since $\mathbb{O}'$ must contain suitable POIs that satisfy all users (or at least a set of them). We are also interested in dynamic environments where both the POIs and users could potentially be on the move and context conditions can change quickly over time. Finally, we also intend to consider other spatial distance calculation approaches, such as heuristic searches (e.g., by using $A^\star$ algorithms), as well as other approaches to compute term distances (e.g., a word embedding approach such as word2vec).

Acknowledgments

Work supported by the project TIN2016-78011-C4-3-R (AEI/FEDER, UE) and the Government of Aragon (Group Reference T35_17D, COSMOS group) and co-funded with Feder 2014-2020 “Construyendo Europa desde Aragon”.

REFERENCES

[1] Xin Cao, Gao Cong, Tao Guo, Christian S Jensen, and Beng Chin Ooi. 2015. Efficient processing of spatial group keyword queries. *ACM Transactions on Database Systems (TODS)* 40, 2 (2015), 13:1–13:48.
[2] Xin Cao, Gao Cong, Christian S Jensen, and Beng Chin Ooi. 2011. Collective spatial keyword querying. In *2011 ACM SIGMOD Int. Conference on Management of Data*. ACM, 373–384.
[3] Harry Kai-Ho Chan, Cheng Long, and Raymond Chi-Wing Wong. 2018. On generalizing collective spatial keyword queries. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1712–1726.
[4] Gao Cong, Christian S Jensen, and Dingming Wu. 2009. Efficient retrieval of the top-k most relevant spatial web objects. *Proceedings of the VLDB Endowment* 2, 1 (2009), 337–348.
[5] María del Carmen Rodríguez-Hernández, Sergio Ilarri, Ramón Hermoso, and Raquel Trillo-Lado. 2017. DataGenCARS: A Generator of Synthetic Data for the Evaluation of Context-Aware Recommendation Systems. *Pervasive and Mobile Computing* 38, Part 2 (2017), 516–541.
[6] María del Carmen Rodríguez-Hernández, Sergio Ilarri, Raquel Trillo-Lado, and Ramón Hermoso. 2015. Location-Aware Recommendation Systems: Where We Are and Where We Recommend to Go. In *Int. Workshop on Location-Aware Recommendations (LocalRec)*, Vol. 1405. CEUR Workshop Proceedings, 1–8.
[7] Yunjun Gao, Jingwen Zhao, Baihua Zheng, and Gang Chen. 2015. Efficient collective spatial keyword query processing on road networks. *IEEE Transactions on Intelligent Transportation Systems* 17, 2 (2015), 469–480.
[8] Claudia Leacock and Martin Chodorow. 1998. Combining local context and WordNet similarity for word sense identification. *WordNet: An electronic lexical database* 49, 2 (1998), 265–283.
[9] Yuhua Li, Zuhair A Bandar, and David McLean. 2003. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering* 15, 4 (2003), 871–882.
[10] Cheng Long, Raymond Chi-Wing Wong, Ke Wang, and Ada Wai-Chee Fu. 2013. Collective spatial keyword queries: a distance owner-driven approach. In *2013 ACM SIGMOD Int. Conference on Management of Data*. ACM, 689–700.
[11] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook*. Springer.
[12] Pengfei Zhang, Huaizhong Lin, Bin Yao, and Dongming Lu. 2017. Level-aware collective spatial keyword queries. *Information Sciences* 378 (2017), 194–214.