Universidade Federal da Bahia
Instituto de Matemática

Programa de Pós-Graduação em Ciência da Computação

EXPLOITING OPEN DATA FOR IMPROVING SPATIAL KEYWORD QUERIES APPLICATIONS

João Paulo Dias de Almeida

QUALIFICAÇÃO DE DOUTORADO

Salvador
1 de julho de 2019

JOÃO PAULO DIAS DE ALMEIDA

EXPLOITING OPEN DATA FOR IMPROVING SPATIAL KEYWORD QUERIES APPLICATIONS

Esta Qualificação de Doutorado foi apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Orientador: Frederico Araújo Durão

Salvador
1 de julho de 2019

RESUMO

to do

Palavras-chave: consulta espacial, linked data, LOD, personalização

ABSTRACT

to do

Keywords: spatial query, linked data, LOD, personalization

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTA DE SIGLAS

PGCOMP Programa de Pós-Graduação em Ciência da Computação 44
PGCA Programa de Pós-Graduação em Computação Aplicada 44

Chapter 1

INTRODUCTION

Chapter 2

DATABASE QUERIES

This chapter introduces core concepts underlying the topics discussed in this research. We discuss textual queries followed by spatial queries and indexes. After that, we present preference query and its extensions.

2.1 TEXTUAL QUERIES

Textual query is a key technology to search engines (ZOBEL; MOFFAT, 2006). A user can type one or more keywords in a textual query to describe the document she wants to retrieve (MANNING et al., 2008). This query searches and retrieves information from textual collections, returning documents relevant to the user that matches the keyword queries (SALMINEN; TOMPA, 1994). Web search engines and desktop search systems are examples of daily applications which employ textual queries.

A textual database is a collection of textual data like web pages, encyclopedias, academic publications, or e-mails. Each element from a textual database is called *document*. Accordingly to Manning et al. (MANNING et al., 2008), Document is any unit which is chosen to build a Retrieval System. In a typical Textual Information Retrieval System, a user describes the document she desires using a set of keywords (also known as "bag of words") (ZOBEL; MOFFAT, 2006).

For example, based on the textual database described in Figure 2.1, a user can identify a document she is interested in posing a textual query. In this scenario, the system considers each line as a document. Therefore, when a user types a set of keywords in a textual query, this query returns all documents inside the textual database that are relevant to the query keywords. As an example, whether the user types the keyword *big*, the textual query returns the documents 2 and 3, because they contain the keyword.

2.1.1 Pre-processing

Inverted Files (IF) are commonly used to process textual queries efficiently (ZOBEL; MOFFAT, 2006). Create an IF requires to extract the terms from each document in a textual database. For this purpose, a pre-processing stage named *parsing* is applied.

1 The old night keeper keeps the keep in the town
2 In the big old house in the big old gown.
3 The house in the town had the big old keep
4 Where the old night keeper never did sleep.
5 The night keeper keeps the keep in the night
6 And keeps in the dark and sleeps in the light.

Figure 2.1 Textual database *Keeper*. Each text line represents a document. Source: Zobel and Moffat (ZOBEL; MOFFAT, 2006).

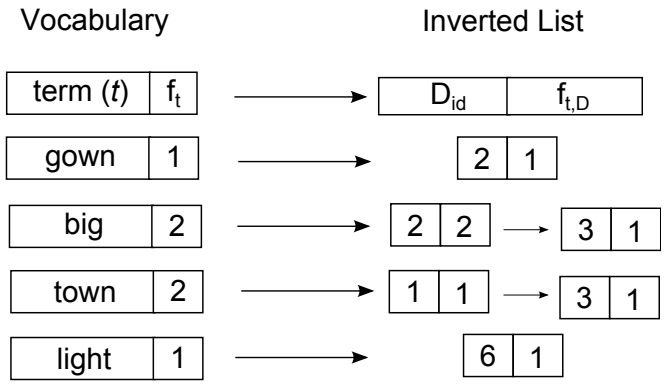

Figure 2.2 Inverted File example using existing terms in textual database *keeper*.

Parsing is realized in two stages: *casefold* and *stop words* removal. The casefold converts every letter in a document to lowercase letters. Applying the casefold in document 1, one obtains "*the old night keeper keeps the keep in the town*" as a result.

Stop words are those that frequently occur in texts or whose function only is to identify a grammar relationship. For this reason, each language has a specific set of stop words. Removing the stop words and applying the casefold, one obtains the following terms from document 1: "*old night keeper keeps keep town*" (ZOBEL; MOFFAT, 2006). Observe that parsing a document reduces the document size considerably, facilitating the storage and organization process.

The IF is composed of the vocabulary (also known as dictionary of terms) and the set of inverted lists (referred as postings list too). Moreover, each term t in a collection has a corresponding inverted list. This list contains an identifier (D_{id}) for each document (D[1]) that contains the term t in its textual description. D_{id} is followed by a integer value representing the frequency $f_{t,D}$ of a term t occurs in a document's textual description. The vocabulary stores a number f_t of documents which contains the term t and a pointer to the inverted list correspondent to the term t (ZOBEL; MOFFAT, 2006).

For instance, Figure 2.2 illustrates a part of a Inverted File (IF) generated from the textual database *keeper* (Figure 2.1). This IF contains the terms *gown*, *big*. *town*, and *light*. The vocabulary stores terms, the number of documents containing the stored terms,

[1]Each line in Figure 2.1 represents a document D while the text inside the line represents the textual description of D.

and a pointer to the inverted list related to the term (represented by the unidirectional arrow). The inverted list stores one tuple for each document which contains a term t, this tuple is composed of the document identifier (D_{id}) and the occurrence frequency($f_{t,D}$) of the term t in D.

2.1.2 Similarity Measure

A Textual Information Retrieval System, which employs textual relevance to retrieve documents, uses a *ranking* to order the possible documents to be presented to the user. In order to create a *ranking*, a similarity measure, or heuristic, is applied to indicate the similarity between the document and the query keywords defined by the user (KELES, 2018; MACKENZIE; CHOUDHURY; CULPEPPER, 2015; ZOBEL; MOFFAT, 2006).

The Information Retrieval community widely use the cosine similarity as an effective formulation of the similarity between a document and a set of query keywords (COHEN; RAVIKUMAR; FIENBERG, 2003; ZHU et al., 2011; ZOBEL; MOFFAT, 2006). Given a textual query T, composed of a set of terms t ($t \in T$), the cosine similarity $\theta(T, D)$ defines the cosine angle, in a n-dimensional space, between the weight vector[2] and the textual description of document D. As a result, a document D is a possible answer to the user only when exists at least one term $t \in T$ which exists in D too ($\exists t \in T : t \in D$).

Zobel and Mofat (ZOBEL; MOFFAT, 2006) propose the following metrics to calculate the cosine between a document and a query:

- the frequency $f_{t,D}$ of term t in the textual description of D
- the frequency $f_{t,T}$ of term t in the query T
- the number f_t of documents containing the term t
- the total number N of documents in the collection

There are many variations of the cosine similarity formulation (MANNING et al., 2008; ROCHA-JUNIOR, 2012). In this thesis, we use the formulation proposed by Zobel and Mofat (ZOBEL; MOFFAT, 2006), presented in Equation 2.1 which employs the metrics described early.

$$\theta(T, D) = \frac{\sum_{t \in T} w_{t,D} \cdot w_{t,T}}{\sqrt{\sum_{t \in D} (w_{t,D})^2 \cdot \sum_{t \in T} (w_{t,T})^2}} \tag{2.1}$$

The term weight t in document D ($w_{t,D}$) is defined by $w_{t,D} = 1 + ln f_{t,D}$, while the weight $w_{t,T}$ of term t in a query T is $w_{t,T} = ln\Big(1 + \frac{N}{f_t}\Big)$. The greater the value of $\theta(T, D)$, the greater is the textual relevance between the document D and the query T. Consequently, $\theta(T, D)$ is also known as the textual score of D related to the query T.

Additionally, $w_{t,T}$ represents a property usually described as *inverse document frequency* (IDF), while $w_{t,D}$ is the *term frequency* (TF). For this reason, the formulation

[2]The weight vector is formed by the terms weight t in T.

described by the Equation 2.1 is also described in the literature as TFxIDF (ZOBEL; MOFFAT, 2006).

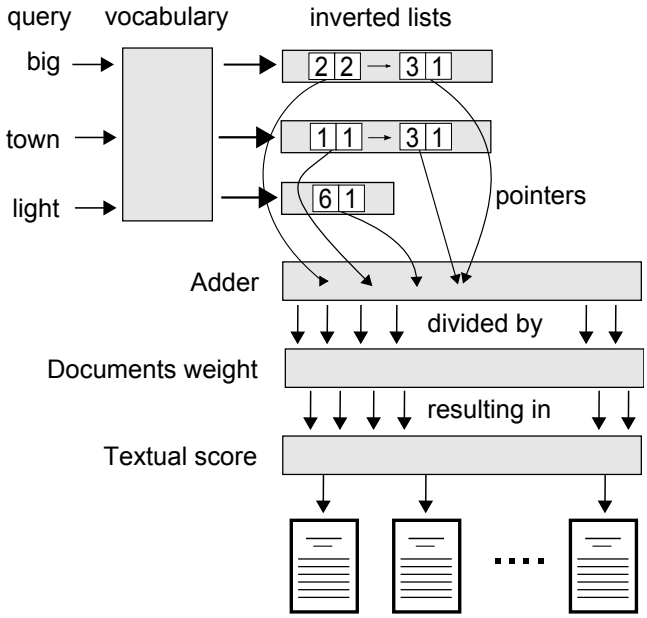

Figure 2.3 Textual query execution example using an Inverted File (IF). Source: Adapted of Zobel and Mofat (ZOBEL; MOFFAT, 2006).

A textual query must generate a *ranking* containing the documents to return to the user. Figure 2.3 exemplifies a textual query processing. Initially, each document has textual score equals zero while a sum array A, of size N, is created to sum the partial textual score of each document. Each array position A_D stores the partial textual score of a document D.

Provided those definitions, each term $t \in T$ contributes $w_{t,D} \cdot w_{t,T}$ (Equation 2.1) to the similarity between a query T and a document D. The A_D position of the sum array (represented by "Adder" in Figure 2.3) stores the summation value obtained from $\sum_{t \in T} w_{t,D} \cdot w_{t,T}$.

The textual score of a document D is the division of its partial score in A_D by the document weight (W_D),

$W_D = \sqrt{\sum_{t \in D} (w_{t,D})^2 \cdot \sum_{t \in T} (w_{t,T})^2}$ (obtained from Equation 2.1).

Last, the documents are ordered by their respective textual scores and then presented to the user.

2.2 SPATIAL QUERIES

Databases adapted themselves to store and organize different types of data efficiently. The spatial data availability associated with technologies advancement made possible a scenario where spatial data is the core of many applications (RIGAUX; SCHOLL; VOISARD, 2001). Today, any individual using a smartphone, is a potential spatial data provider due to the popularization of the Global Positioning System (GPS).

Satellite images, medical equipment, or Geographic Information System (GIS) are other sources of spatial data which provide a large amount of data. Unfortunately, manipulate this volume of data is expensive and impractical to users who don't have proper computational tools. This task becomes even more difficult when is required to analyze the data in details.

2.2.1 Spatial Objects

A spatial database system offers support to spatial objects like points, lines, and polygons (GÜTING, 1994; RIGAUX; SCHOLL; VOISARD, 2001). This system provides additional support to spatial data modeling and spatial queries description. In order to process spatial queries efficiently, the spatial database system employs spatial indexes (GÜTING, 1994).

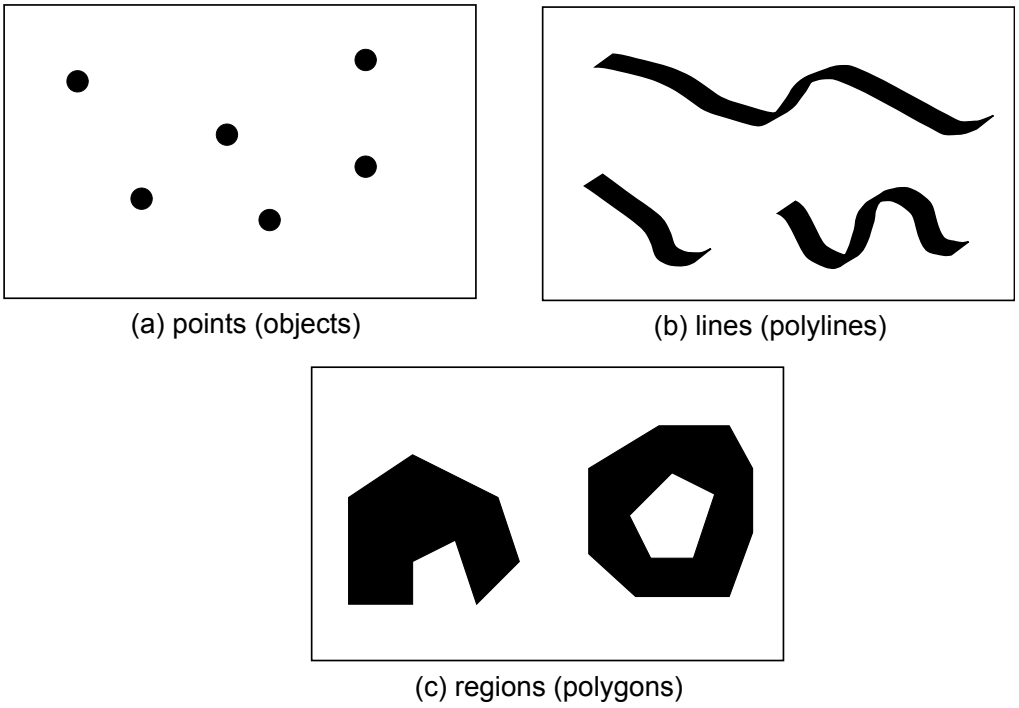

Figure 2.4 Basic spatial objects. Source: Adapted of Rocha-Junior (ROCHA-JUNIOR, 2012).

Spatial objects. Figure 2.4 presents some of the basic spatial objects: points (objects), lines, and regions (polygons). A point (Figure 2.4(a)) represents an object whose area is not relevant, only its spatial location. For instance, a point can represent a reference object location (i.e. restaurant) or a person location. A line (Figure 2.4(b)) can represent a river, a road, or power lines. Besides, it is important to notice that lines can intersect other lines. At long last, a region (Figure 2.4(c)) usually is modeled like a polygon and can describe spatial objects whose spatial area is relevant like a farm or a forest. Regions are disjoint; however, they can have holes or can be composed of many disjoint pieces (GÜTING, 1994; ROCHA-JUNIOR, 2012). In this work, the term "spatial

object" refers to one user's point of interest associated with a textual description or score value.

2.2.2 Range Query

Because of the large volume of spatial data available for search, the popularity of spatial queries increase. Among the most important types of spatial queries employed in spatial databases are the spatial selections based on predicates (GÜTING, 1994; ROCHA-JUNIOR, 2012). Given a database, a spatial selection returns the set of objects which satisfies the predicate. This predicate can be represented by one or more spatial relationships - the most significant operation provided by the spatial algebra (GÜTING, 1994). These spatial relationships can be topological (i.e. adjacency, disjunction), directional (i.e. above, below, to the left), and metric (i.e. distance), among others. The sentence "*find all restaurant in a 100m radius from my actual location*" is an example of spatial selection.

In this work, we direct our focus to one spatial selection in particular: the *range*.

Range. Given the query location $q.l$ and the distance $dist(p, q.l)$ (euclidian distance between $q.l$ and an object p) , the spatial query *range* retrieve all p objects whose distance values are smaller than the radius r, $dist(p, q.l) \leq r$ (ROCHA-JUNIOR, 2012; YIU et al., 2007). Therefore, r defines the query's spatial neighborhood.

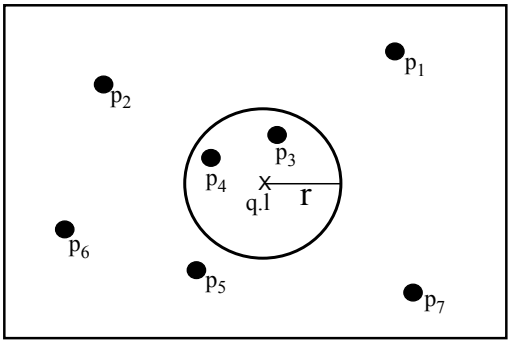

Figure 2.5 Spatial selection example: range. Source: Rocha-Junior (ROCHA-JUNIOR, 2012).

Figure 2.5 illustrates a range query where $q.l$ is the query location and r is the interest radius. Processing this query on the spatial area illustrated in Figure 2.5 returns the points p_3 and p_4 as result.

2.2.3 Spatial Indexes

A spatial database system requires a mechanism to improve the spatial objects retrievement, taking in consideration their locations and the user's need (GUTTMAN, 1984). In order to assist in this task, several researchers proposed many spatial indexes (BECKMANN et al., 1990; GUTTMAN, 1984; PAPADIAS et al., 2001; SAMET, 1984). We present some of these spatial indexes in this subsection.

The R-tree is a balanced tree, almost identical to a B-tree (BARUFFOLO, 1999; BAYER; MCCREIGHT, 1970; COMER, 1979) whose leaves have pointers to space-textual objects. R-tree is dynamic; hence, insertion and removal of elements can be performed in conjunction with queries without having to reorganize the tree periodically (GUTTMAN, 1984). In addition, R-tree nodes are generally the size of a disk page, and their structure is designed to search only a small number of nodes. Thus, each node of the R-tree has a minimum and a maximum number of entries (GUTTMAN, 1984; ROCHA-JUNIOR, 2012).

There are two types of nodes in an R-tree: intermediate nodes and leaf nodes. The intermediate node contains pointers to the descendant nodes, while the leaf nodes have pointers to the indexed objects. The entries of an R-tree are formed by (MBR, *id*). Minimum Bounding Rectangle (MBR) is an n-dimensional rectangle surrounding the indexed object, and *id* is a number that identifies the input. The *id* of an intermediate node is a pointer (address) to another node in the tree (descendant node), while the MBR of an intermediate entry involves the MBRs of all entries in the child node. In the input of a leaf node, *id* is the identification of the object in the database and the MBR is the smallest possible n-dimensional rectangle that can wrap the indexed spatial object (ROCHA-JUNIOR, 2012).

Figure 2.6(a) is the representation of a spatial area where objects (p) are indexed in a R-tree. Under those circumstances, $q.l$ is the query location, r defines the spatial neighborhood of $q.l$, and m_1, m_2, m_3, and *root* are the MBRs. On the side, in Figure 2.6(b), the root is a intermediate node that has three intermediate entries m_1, m_2, m_3 which point to the leaf nodes n_1, n_2, n_3, respectively. The intermediate entry $(m_1, *n_1)$ contains the MBR m_1 that involves all stored objects in node n_1, and a pointer $*n_1$ pointing to the node n_1. The leaf node n_1 contains two leaf entries: $(m_{p1}, *p_1)$ and $(m_{p3}, *p_3)$, where m_{p1} is the MBR involving the spatial object p_1 and $*p_1$ is the pointer (identifier) to object p_1 in the database.

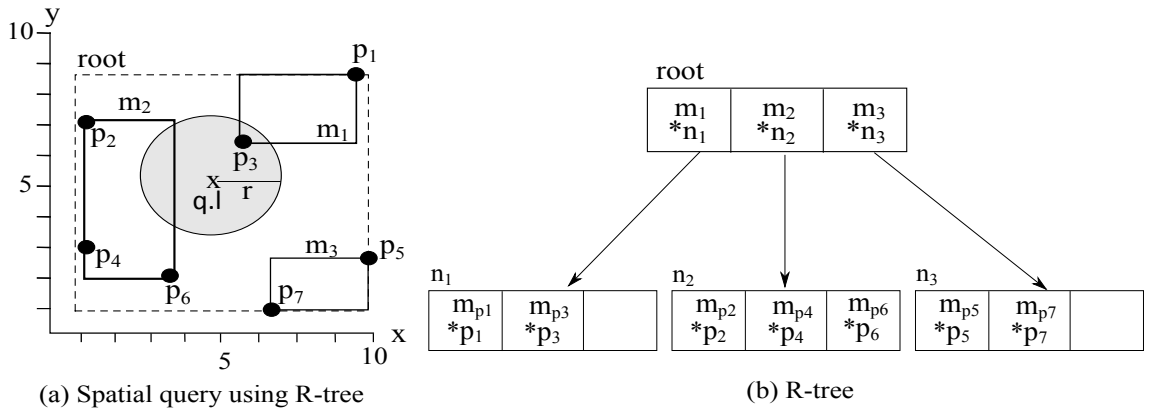

Figure 2.6 R-tree examples. Adapted of Rocha-Junior (ROCHA-JUNIOR, 2012).

For example, Figure 2.6(a) presents a range query processed with a R-tree. The query searches for spatial objects inside the spatial neighborhood defined by r. In other words, the query searches for objects inside the circumference which has 'x' as the center and r as

the radius. The range query starts the search in the root and then searches the entries, verifying which entry has a MBR distance to $q.l$ smaller than the size of r. Knowing that p is the nearest point in a MBR to $q.l$, $dist(p, q.l)$ defines the shortest distance of a MBR to $q.l$, this way $dist(p, q.l)$ have to be lower than r to the entry be visited. In Figure 2.6(a), two entries satisfy this condition: $(m_1, *n_1)$ e $(m_2, *n_2)$. As a result, the leaf nodes n_1 and n_2 are accessed to search for the leaf entries whose MBR[3] is inside the spatial neighborhood defined by r, returning object p_3 as consequence.

The R-tree is based on a heuristic optimization, consisting in minimize the MBR area of each intermediate node. However, this criterion proved not to be the best (BECKMANN et al., 1990). One of the most well-known variations of the R-tree is the R*-tree (CHEN et al., 2013; HARIHARAN et al., 2007; WU et al., 2012; ZHOU et al., 2005). The R*-tree is superior to the R-tree in query processing and in the algorithm that defines the MBR of the nodes (BECKMANN et al., 1990).

The R*-tree reduces the coverage area of the MBRs involving intermediate nodes. Thus, fewer tree branches are used during query processing, resulting in less access to disk pages. In addition, R*-tree reduces the overlap between MBRs, reducing the probability of having more than one MBR covering the same area and increasing the efficiency of the query (ROCHA-JUNIOR, 2012).

Another widely used variation of R-tree is the aggregate R-tree (aR-tree), proposed by (PAPADIAS et al., 2001). The main feature of aR-tree is to use pre-aggregated non-spatial data to optimize query processing. In other words, each node of an aR-tree has a non-spatial data (eg, a numeric value) added.

For instance, assume that each object p in Figure 2.6 has a non-spatial score (numeric value). In this context, a query can be made to search for objects in the spatial neighborhood defined by r and that have a score greater than 0.7. In a traditional R-tree, this query needs to be performed in two steps. Initially, all objects that are in the spatial neighborhood of $q.l$ are selected. Then the score of each selected object is checked, and only those that have a score greater than 0.7 are returned (ROCHA-JUNIOR, 2012; PAPADIAS et al., 2001).

In order to optimize this process, each intermediate node of aR-tree stores a value that is obtained by an aggregated function applied to the child node inputs. Under those circumstances, the Max aR-tree is used because it employs the aggregated function $max()$. Thus, the maximum value of the score on the child nodes is added to their respective intermediate node (ROCHA-JUNIOR, 2012; PAPADIAS et al., 2001).

Figure 2.7 represents a Max aR-tree where the aggregated function $max()$ was applied. Therefore, it is observed that the score stored at the intermediate input $(m_1, 0, 9, *n_1)$ is 0.9 because this is the highest score value between the entries of the node n_1: $(m_{p1}, 0, 5, *p_1)$ e $(m_{p3}, 0, 9, *p_3)$. The structure and the way the aR-tree query is executed are similar to that of the R-tree. However, only entries that satisfy the spatial and non-spatial conditions are visited. For example, to find objects that are in the spatial neighborhood of $q.l$ and have a score greater than 0.7, the root is accessed for the input that satisfies these

[3]In this case, the leaf entries are bi-dimensional points. Thereby, the upper right vertex is identical to the lower left vertex.

two conditions (neighborhood criterion and score). Thus, only the input $(m_1, 0, 9, *n_1)$ is visited, and the object p_3 is returned because it is the only one that has a score greater than 0.7 and has a distance to $q.l$ lower than the size of r.

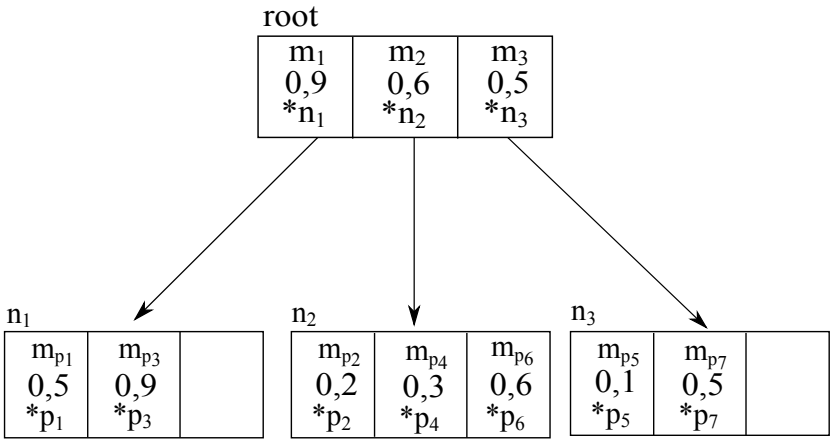

Figure 2.7 AR-tree example.

2.3 PREFERENCE QUERIES

Databases provide a rigid way to define the characteristics of the retrieved data while using traditional queries (LACROIX; LAVENCY, 1987). The lack of flexibility culminate in a very large, or very small, set of retrieved data. Therefore, current Information Systems employ techniques to describe and process user preferences (CHOMICKI, 2003). These preferences are a important tool to filter the information, reducing the data volume presented to the user (CHOMICKI, 2003).

In detail, Table 2.1 illustrates a dataset H which contains information about hotels, such as their respective daily price; and the distance from these hotels to the beach. Assume a user who does not know the dataset and wants to find a cheap hotel. Using traditional queries, the user can request to list the hotels with daily price below 60. In this case, no hotel will be returned. In contrast, if the user requests the list of hotels with rate higher than 60, the complete dataset will be returned. In this way, the user will have to go through the database until she finds the hotel she wants, making it difficult to find the cheapest hotel.

Preference Queries (LACROIX; LAVENCY, 1987) allow the user to express their preferences more clearly and accurately[4]. One can solve the problem described above by setting the query as follows: "select hotels with the lowest price values, stop after k" (ROCHA-JUNIOR, 2012). Thus, by considering $k = 3$ and using the data from Table 2.1, this preference query returns the hotels h_2, h_4, and h_5.

[4]Borzsony, Kossmann and Stocker (BORZSONY; KOSSMANN; STOCKER, 2001) demonstrate how to implement a preference query using SQL (without making modifications in the database system). They discuss the reasons to such an implementation presents poor performance when compared to an implementation of the preference query using an extension of a database system with a new logical operator representing the preference query.

Table 2.1 Dataset example with hotels. Each object (hotel) contains two attributes: *price* (daily price in US dollars) and `distance` (distance to the beach in meters). Source: Adapted of Rocha-Junior (ROCHA-JUNIOR, 2012).

Hotel	**Price** ($)	**Distance** (m)
h_1	300	50
h_2	100	100
h_3	500	100
h_4	90	300
h_5	250	500

Preference queries are classified by the methods employed to express the users' information need. The *qualitative* preference query specifies the user preference directly between pairs of objects (tuples) in the database, using a preference formula $f(a, b)$. Given two objects h_1 and h_2 in dataset H, the preference formula $f(h_1, h_2)$ determines whether an object satisfies the user's needs. The preference formula $f(h_1, h_2)$ is a binary operation between objects h_1 and h_2. Thus, when the result of this formula is *true*, the query considers object h_1 satisfy the user's needs better than the object h_2. The preference formula is defined using logical operators (CHOMICKI, 2003; KIESSLING, 2002).

For example, consider the database described by Table 2.1 and a user interested in the cheapest and closest to the beach hotel. This user interest can be described by the preference formula $f_1(a, b) = [(a[\text{rate}] \leq b[\text{rate}]) \wedge (a[\text{distance}] \leq b[\text{distance}]$. In Table 2.1, the object h_2 satisfies the user's better than the object h_3. Both hotels have the same distance to the beach, however the hotel h_2 is cheaper. In this scenario, we say that h_3 is dominated by h_2 since $f_1(h_2, h_3) = true$. All not dominated objects are valid responses to the query described by the formula $f_1(a, b)$.

On the other hand, the *quantitative* preference query specifies the preference indirectly for each object in the data set. A score function evaluates the attributes of an object, producing a numeric value (score) that represents the importance of this object to the user's needs. Quantitative queries are often referred to as *top-k queries*. This type of query requires a function to calculate the objects' score and the number of objects (k) to return from the database (ROCHA-JUNIOR, 2012).

For instance, in the dataset H presented in Table 2.1,the hotel h_1 can be represented by $h_1 = \{300, 50\}$, where the value 300 is positioned in column (dimension) 1 of the table and the value 50 in column 2. One can use the quantitative preference query to find the three cheapest and closest hotels to the beach. Assuming a score function $f(h) = 0.5 * h[\text{rate}] + 0,5 * h[\text{distance}]$, objects with lower scores are those closer to the user's need. Thus, the scoring function returns the score values $f(h_2) = 100$, $f(h_1)$=*175*, and $f(h_3)$=*195* for the objects h_2, h_1, and h_3, respectively. Therefore, the quantitative preference query returns the objects h_2, h_1 e h_3 as response.

Most top-k query processing techniques use scoring functions called monotonic functions, since these functions have special properties that allow efficient processing of top-k query (ILYAS; BESKALES; SOLIMAN, 2008). Consider an object $h \in H$ represented by

$h = h[1], ..., h[n]$, where $h[i]$ is a numerical value in the i dimension. A function f_h defined on the attributes (dimensions) of an object h is monotonic, if for all objects $h, q \in H$, $f_h \leq f_q$ when $h[i] \leq q[i]$ for all i (ROCHA-JUNIOR, 2012). To demonstrate, the function $f(h) = 0,5 * h[\text{price}] + 0,5 * h[\text{distance}]$ would be considered monotonous if for every object $h_x, h_y \in H$, $f(h_x) \leq f(h_y)$ when $h_x[i] \leq h_y[i]$. Since $f(h_2) \leq f(h_1)$ ($f(h_2) = 100$, and $f(h_1) = 175$) but $h_2[2] > h_1[2]$ ($h_2[distance] = 100$, and $h_1[distance] = 50$), this function is not monotonic.

2.4 SPATIAL PREFERENCE QUERIES

Spatial databases manage large collections of geographic entities. Each entity has geographic coordinates which indicate the position of the object in space. Moreover, it is common to associate non-spatial information to the geographic coordinates such as textual description, object name, size, or price of the object (YIU et al., 2007).

Top-k spatial queries return a set of spatial objects (geographic entities) that can serve the user's need. However, each query defines its own set of parameters to represent the user preference. Yiu et al. (YIU et al., 2007) present a new type of query top-k, the Spatial Preference Query. In this query, the k best user's interesting objects are defined through the quality of features[5] in the spatial neighborhood of each interesting object.

Thereby, given a set of interesting objects P (e.g., candidate locations), the Spatial Preference Query returns the k objects in P with the highest scores. The score of an interesting object is defined by the quality of the feature (e.g., cafes, restaurants, hospitals) in its neighborhood. In this fashion, the feature quality can be obtained through an online ranking system, such as Booking[6] or Foursquare[7] where users evaluate various types of features (YIU et al., 2007).

For example, the white points p in Figure 2.8 represent interesting objects. In addition, the gray dots represent restaurants while the black dots represent cafeterias. Each restaurant and cafeteria (black and gray dots) has a predefined score value, represented by the real number positioned around each of these points. The bigger the feature score, the higher the feature quality.

Assuming a tourist wants to get the best hotels in terms of cafeterias and restaurants, the Spatial Preference Query returns the interesting objects (hotels) with the highest scores. In other words, the tourist is interested in the hotel p that maximizes the score $\tau(p)$, defined as the sum of maximum restaurant quality and maximum cafeteria quality in the neighborhood of p (i.e., the dotted circle at p with a 0.2 km radius). Thus, the interesting objects score values for this range query are $\tau(p_1) = 0.7 + 0.5 = 1.2$, $\tau(p_2) = 0.9 + 0 = 0.9$, and $\tau(p_3) = 0.4 + 0 = 0.4$. Interesting objects that do not have cafeterias or restaurants in their neighborhood receive the value zero as the score, situation represented by objects p_2 and p_3. As can be seen, we obtain the object p_1 as the top-1 result of the range Spatial Preference query (YIU et al., 2007).

[5] Consider "feature" as a class of objects in a spatial map, such as a specific installation or a service. Each feature is associated with a score that is predefined by a classification system.

[6] www.booking.com

[7] www.foursquare.com

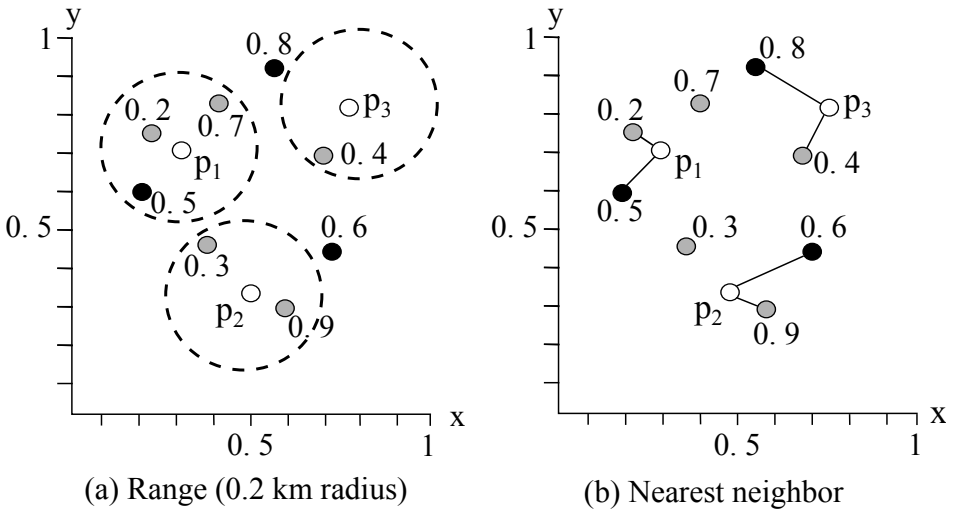

Figure 2.8 Spatial Preference queries examples using different ways to define the spatial neighborhood of an object of interest. Source: Yiu et al. (YIU et al., 2007).

Likewise, Figure 2.8 (b) illustrates the scenario where the score $\tau(p)$ of a hotel is taken as the sum scores of its nearest restaurant and cafeteria (indicated by connecting line segments). Therefore, we have $\tau(p_1) = 0.2 + 0.5 = 0.7$, $\tau(p_2) = 0.9 + 0.6 = 1.5$, $\tau(p_3) = 0.4 + 0.8 = 1.2$, resulting in p_2 as the best hotel (YIU et al., 2007).

Generally speaking, the Spatial Preference Query uses two steps to select interesting objects. First, it calculates the distance of the interesting object for a given feature. Then, it orders the objects by an aggregation function on their scores (YIU et al., 2007).

Besides this Top-k query, a lot of work is being developed in this research area (LI et al., 2018; SHANBHAG; PIRK; MADDEN, 2018; CARMEL; GUETA; BORTNIKOV, 2018; MENG; ZHANG; ZHAO, 2018), demonstrating the popularity of Top-k queries.

2.5 TOP-K SPATIAL KEYWORD QUERY

Among spatial queries, there those that use keywords to express the user's information need. In this section, we will describe some queries that use this model to retrieve the desired information. Then, we discuss hybrid indexes capable of simultaneously indexing spatial and textual data. These spatio-textual indexes aim to support efficient processing of queries which access data with spatial and textual properties.

Given a spatial location and a set of keywords, a *top-k Spatial Keyword* query (SK) (CAO et al., 2012; CHEN et al., 2013) returns objects that are spatially close to the user's location and textually relevant to the keywords. All returned objects have these two characteristics: user proximity and textual relevance. A score function evaluates the spatial proximity between an object and the user, as well as the textual relevance of the object description considering the set of keywords. The response of this query is ordered from the score values generated for each object by the scoring function.

Suppose a user wants to find a bar where they have samba presentation, in the spatial area described by Figure 2.9. This user poses a top-3 spatial keyword query q with the

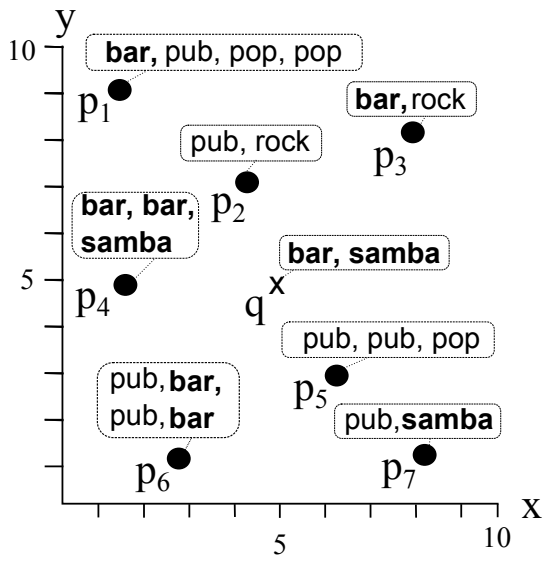

Figure 2.9 Spatial area containing bars and pubs. Source: Rocha-Junior et al. (ROCHA-JUNIOR et al., 2011).

following keywords: "samba" and "bar". The user location is the same query location q in Figure 2.9.

In this example, the top-k Spatial Keyword query returns a ordered set containing the objects p_4, p_6, p_7. The object p_4 is the top-1 result because its textual description is similar to the keywords provided by the user, and it is the object closest to the query location. Following, object p_6 is the top-2 result, since the textual description of p_6 is more relevant than that of p_7, and p_6 also gets closer to q than p_7.

2.5.1 Spatio-textual Indexes

Many applications now use a large amount of spatial data, such as Twitter[8] and Flickr[9]. These applications can benefit from Spatial Keyword Query (SK) and other spatio-textual queries, but the cost of processing these queries is prohibitive (ROCHA-JUNIOR et al., 2011). For this reason, spatio-textual indexes play an important role in the processing of these queries. These indexes store data that contains textual and geographic information, enabling efficient processing of spatio-textual queries (CHEN et al., 2013).

Spatial-Keyword Inverted File (SKIF) proposed by Khodaei, Shahabi and Li (KHODAEI; SHAHABI; LI, 2010) is an Inverted File (IF) capable of indexing and searching for spatial and textual data in an integrated way, using only one structure to manage the two parts of data simultaneously. Figure 2.10 illustrates a SKIF where the space is partitioned into cells. SKIF represents the keywords as these cells.

Similar to IF, SKIF is composed of a vocabulary and a set of inverted lists. The vocabulary contains all terms in objects' textual descriptions and identifiers for the cells that constitute the grid over the spatial area of interest. For each distinct term, the

[8] www.twitter.com
[9] www.flickr.com

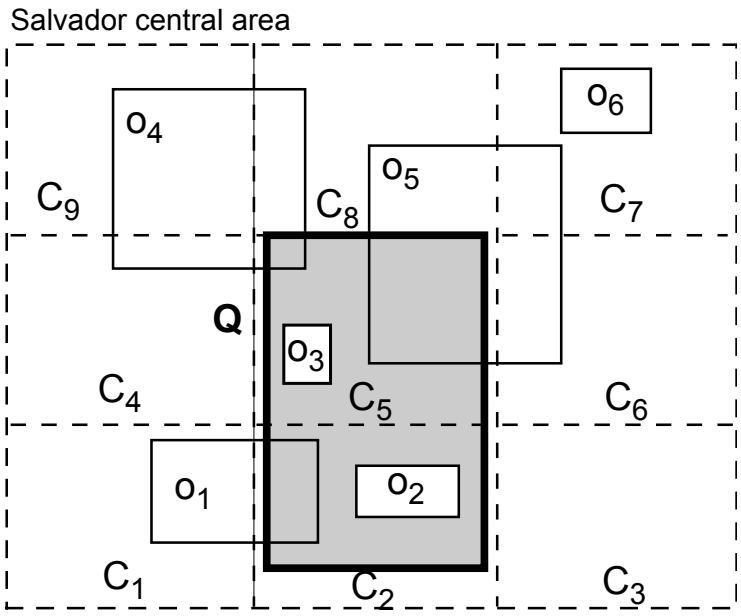

Figure 2.10 The search area is defined by the user (e.g., "central area of Salvador"), then the system divides it into a grid (C_i) to apply the SKIF index. The dark rectangle represents the query location, while light rectangles (d_i) represents objects near the user's search area. Source: adapted of (KHODAEI; SHAHABI; LI, 2010).

following values are stored in the vocabulary: the number of objects o_t containing the term t, an inverted list pointer, and the indexed term type. Each term t has a corresponding inverted list, where is stored the objects identifiers that have the term t and the normalized frequency with which term t appears in each object description (KHODAEI; SHAHABI; LI, 2010).

SKIF is designed to process a query capable of returning the k objects that have the highest textual and spatial scores concomitantly. However, SKIF process the location of each object as a region rather than a point. Spatial relevance is expressed by the overlap between the query region (dark region) and the region of an object (o_i) (CHEN et al., 2013). Therefore, despite being a hybrid index, SKIF is not able to process a top-k Spatial Keyword query because it does not consider the query location as a point. Even though Chen et al. (CHEN et al., 2013) modified SKIF to process Boolean Range queries (BRQ), they report not be able to identify a way to process top-k queries using SKIF.

Together with SKIF, many other cell-based structures have already been proposed to process spatio-textual objects (BENTLEY; FRIEDMAN, 1979; GUTTMAN; STONEBRAKER, 1982). We discuss below other hybrid indexes capable of indexing spatio-textual objects. These indexes employ trees such as the R-tree proposed by Guttman (GUTTMAN, 1984), and the R*-tree proposed by Beckmann et al. (BECKMANN et al., 1990).

Cong, Jensen e Wu (CONG; JENSEN; WU, 2009) incorporate document similarity to propose a new spatio-textual index called DIR-tree (Document-similarity enhanced Inverted file R-tree). DIR-tree combines spatial and text information during the index

building process, keeping at the same level of the tree objects that are spatially close to each other. In addition, it maximizes the textual similarity between objects that are part of the same MBR (CONG; JENSEN; WU, 2009; WU; CONG; JENSEN, 2012). A β parameter is introduced to determine the weight between the spatial and textual part of the data. For example, depending on the β value, the objects of the same MBR can be close but with little textual relevance to each other.

Each DIR-tree node is associated with an Inverted File (IF), and each leaf node contains an object summary that provides textual information about the objects (LI et al., 2011). Figure 2.11 presents spatial objects O_i and their respective MBRs R_i. Coupled with Figure 2.12 that exemplifies the organization of a DIR-tree to index the objects in Figure 2.11

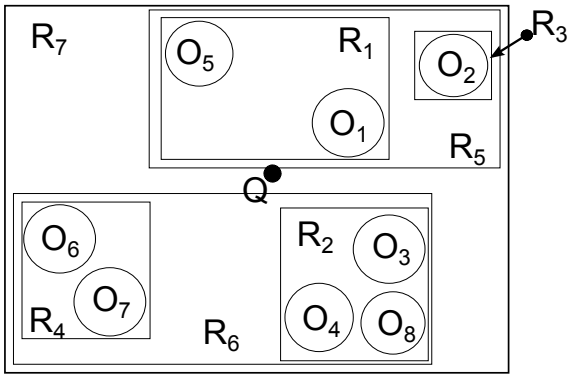

Figure 2.11 Spatial objects O_i and their respective MBRs R_i. Source: (CONG; JENSEN; WU, 2009).

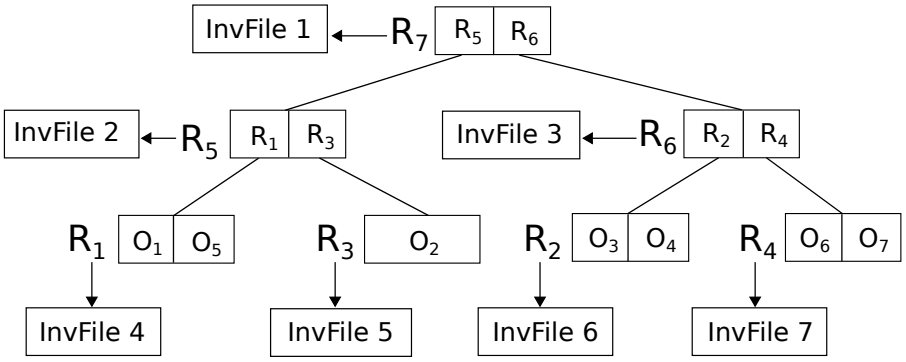

Figure 2.12 DIR-tree structure. Source: (CONG; JENSEN; WU, 2009).

To demonstrate, consider a spatio-textual query receiving a set of keywords T. Assume that the object O_1 (Figure 2.11) is the most textually relevant for the query. Hence, to process this query Q, it is necessary to traverse only the nodes R_7, R_5, and R_1 of the tree shown in Figure 2.12, to reach the object O_1 and return it as query response.

Alternatively, Rocha-Junior et al. (ROCHA-JUNIOR et al., 2011) propose a new structure for indexing spatio-textual data, called Spatial Inverted Index (S2I). This struc-

ture optimizes the processing of the top-k Spatial Keyword query. S2I is similar to the Inverted File (ZOBEL; MOFFAT, 2006; ROCHA-JUNIOR et al., 2011), but it stores the most frequent terms of the collection with a different method. S2I maps the most frequent terms of the collection to aggregated R-trees (aR-tree) (PAPADIAS et al., 2001), where each tree stores only objects that have the same term t. In like manner, S2I stores the less frequent terms in file blocks, where each block stores objects that have the same term t.

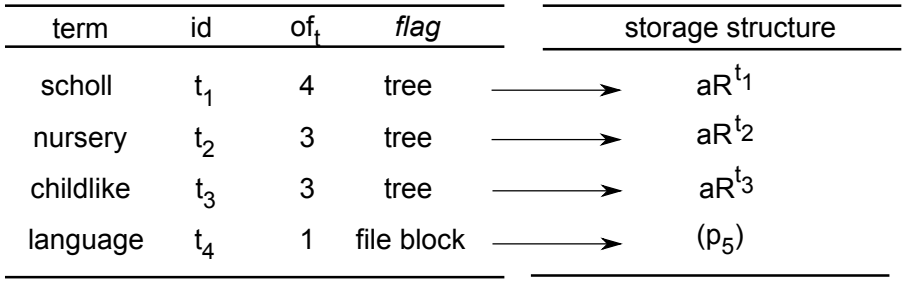

Figure 2.13 *Spatial Inverted Index.*

The S2I (exemplified in Figure 2.13) is composed of vocabulary, file blocks (b_i) and aR-tree's (aR^{t_i}). The vocabulary stores each distinct term in the database (e.g., "school", and "nursery"). For each term t_i, it stores the amount of objects of_t in which t_i occurs. Also, it stores a flag indicating what type of structure the term is stored in (block or tree), and a pointer to the structure containing the term (represented by the unidirectional arrow in Figure 2.13).

Each block file stores a set of objects. For each object in this set, it stores the object's identification $o.id$, the object location $o.l$ and the frequency $f_{d,t}$ with which the term t occurs in the textual description of this object.

The leaf nodes of the aR-tree store the same information as the file blocks: $o.id$, $o.l$ e $f_{d,t}$. The intermediate nodes store a Minimal Bounding Rectangle (MBR) that involves the spatial location of all objects that are in the subtree. The intermediate node also stores a non-spatial value, representing the maximum value of $f_{d,t}$ of the objects stored in the subtree (PAPADIAS et al., 2001). Thus, objects can be accessed decreasingly by $f_{d,t}$ values, and spatial proximity (ROCHA-JUNIOR et al., 2011).

According to Rocha-Junior et al. (ROCHA-JUNIOR et al., 2011), the results obtained using S2I demonstrate the cost optimization of the query, as well as the cost to update an existing term in the collection. For queries with only one keyword, S2I traverses only a small tree or file block. When the query has several keywords, it is necessary to go through only a set of small trees or blocks of files, dispensing access to an external inverted index.

Chapter 3

LINKED OPEN DATA

The Web has evolved into a space where both documents and data are linked (BIZER; HEATH; BERNERS-LEE, 2009). In order to support this new Web, a set of practices for publishing and connect structured data has been proposed by Berners-Lee (BERNERS-LEE, 2011). This set of practices is known as Linked Data because it enables a user to start browsing in one data source and then navigate along with links into related data sources. In addition, Linked Data is published in such a way that the data is machine-readable, enabling new possibilities for applications. Berners-Lee (BERNERS-LEE, 2011) defines the following set of practices to create Linked Data:

1. Use Uniform Resource Identifiers (URIs) as name for things;
2. Use HTTP URIs to publish your data;
3. Provide useful information using the standards (RDF, SPARQL);
4. Include links to other URIs, so users can discover more things.

In a nutshell, Linked Data relies on these three technologies: Uniform Resource Identifiers (URIs) (BERNERS-LEE; FIELDING; MASINTER, 2005), the HyperText Transfer Protocol (HTTP) (FIELDING et al., 1999), and the Resource Description Framework (RDF) model. A simple way to create linked data is using one RDF file with a URI which points into another file. Suppose an RDF file, named <http://example.org/Hotels>, where hotels around the world are described. Local identifiers (#Venice, #Italy and #Hotel_Danieli) are used to describe one hotel (resource). In Listing 3.1, hotel Danieli is described with RDF. An HTTP URI <http://example.org/Hotels/#Hotel_Danieli> can be assigned, enabling anyone on the Web to access the hotel's description. When this data is released under an open license it is called Linked Open Data (LOD). In this work, we use two LOD sources: DBpedia and LinkedGeoData.

```
<rdf:Description about="#Hotels_in_Venice"
  <rdf:type rdf:Resource="http://example.org/Hotels/#Hotel_Danieli">
</rdf:Description>
```

Listing 3.2 Description of hotels in Venice in a RDF file

```
<rdf:Description about="#Hotel_Danieli"
  <rdf:type rdf:Resource="#Italy">
  <rdf:type rdf:Resource="#Venice">
</rdf:Description>
```

Listing 3.1 Description of hotel Danieli in an RDF file

Now, suppose there is another RDF file (listing 3.2) containing the description of Hotels in Venice. Hotel Danieli is in Venice, however, is not required to describe it again in this new file. Hotel Danieli is described by his HTTP URI which points to his description. When this data is released under an open license it is called Linked Open Data (LOD). In this work, we use two LOD sources: DBpedia and LinkedGeoData.

In this work, we use two LOD sources: DBpedia and LinkedGeoData. The DBpedia dataset has derived its data corpus from Wikipedia. Wikipedia is heavily visited and under constant revision. In turn, LinkedGeoData uses the information collected by the OpenStreetMap (OSM) project and makes it available. OSM is a collaborative editable map, where data is collected by volunteers performing systematic ground surveys using tools such as a handheld GPS.

3.1 RESOURCE DESCRIPTION FRAMEWORK - RDF

The Resource Description Framework (RDF) is a framework for representing information in the Web. It has an abstract syntax and formal semantics which enable deductions in RDF data.

RDF represents information in a minimalist and flexible way. RDF usually shares information between applications which have individually design setups. This framework increases the value of information as it becomes accessible to more applications across the entire Internet (KLYNE; CARROLL, 2006).

3.1.1 Graph Data Model

The RDF structure is a collection of triples which each contains a subject, a predicate, and an object. A set of such triples is called an RDF graph. Figure 3.1 illustrates an RDF graph using a node and directed-arc diagram. In this graph, each triple is represented as a node-arc-node link (for this reason the term "graph" is employed) (KLYNE; CARROLL, 2006; PAN, 2009).

Each triple expresses a statement of a relationship between the nodes that it links. Each triple has three parts:

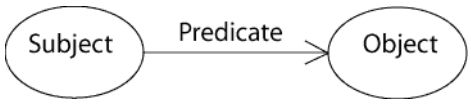

Figure 3.1 RDF Graph.

1. a subject,
2. an object, and
3. a predicate (also known as property) that denotes a relationship.

The nodes of an RDF graph are its subjects and objects, while the arc always points toward the object.

An RDF triple denotes that some relationship, indicated by the predicate, holds between the things denoted by subject and object of the triple. According to (KLYNE; CARROLL, 2006), the assertion of an RDF graph amounts to asserting all the triples in it. Therefore, the meaning of an RDF graph is the conjunction (logical AND) of the statements corresponding to all the triples it contains.

3.2 SPARQL

SPARQL is a query language that can be used to express queries across diverse data sources. The data queried using SPARQL might be stored natively as RDF or viewed as RDF via middleware. A SPARQL endpoint is used to enable users to query a knowledge base via the SPARQL query language. DBpedia and LinkedGeoData endpoints can be accessed at http://dbpedia.org/snorql/ and http://linkedgeodata.org/sparql. Listing 4.1 introduces a SPARQL query to obtain features within 200 m from an object of interest. In Listing 4.1, *objectURI* is a URI to an object of interest.

SPARQL contains capabilities for querying graph patterns along with their conjunctions and disjunctions. Essentially, a SPARQL query consists of a pattern which is matched against a data source, and the values obtained from this matching are processed to give the answer. The results of SPARQL queries can be result sets or RDF graphs. Listing 3.3 introduces a SPARQL query to obtain objects within 20 km radius of New York City.

```
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT DISTINCT ?resource ?label ?location
WHERE {
    dbr:New_York_City geo:geometry ?sourcegeo.
    ?resource geo:geometry ?location;
    rdfs:label ?label.
    FILTER( bif:st_intersects( ?location, ?sourcegeo, 20 )).
  }
```

Listing 3.3 SPARQL query to obtain objects within 20 km radius of New York city

The predicate *geo:geometry* is defined at Geo-SPARQL (PERRY; HERRING, 2012), an ontology that represents features and geometries. In Listing 4.1, the variable *location* matches with the spatial coordinates of objects around an object of interest. The function *bif:st_ intersects()* returns true if there is at least one point in common between the spatial coordinates *location* and *sourcegeo*. The tolerance for the matching in units of linear distance is supplied at the third parameter of *bif:st_ intersects()*. The tolerance is 200 m as illustrated at Listing 4.1.

Chapter 4

PRELIMINARY PROPOSAL: ENHANCING SPATIAL KEYWORD PREFERENCE QUERY WITH LOD

Some preference queries compare the object's textual description with query keywords to evaluate the object's relevance to the user (CAO et al., 2012; CONG; JENSEN; WU, 2009). In this way, the more words in common, the better the textual relevance between an object and the query keywords. However, this evaluation method has limitations, especially to objects with short textual descriptions. For example, suppose a spatial area (e.g. a city) where two spatial objects are located. The query keywords are "japanese restaurant" and each object has one textual description represented by the following strings: "oriental food" and "cinema". The Spatial Keyword Preference query is not able to return any object because neither the word "japanese" nor "restaurant" is present in any textual description. Note that "oriental food" is related to "japanese restaurant", but the evaluation method is not able to identify this relationship. One possible solution for this problem is offering a wider textual description for the spatial objects. The object described as "oriental food" could be a Japanese restaurant but we can not be sure because of its poor textual description.

Motivated by this problem, we use the data available at Linked Open Data (LOD) cloud to enrich the textual description of objects. A large number of researches have recently studied how to improve the object's textual description using the LOD cloud. This improvement is applied in several areas of research, such as Recommender (HEGDE et al., 2011; FERNÁNDEZ-TOBÍAS et al., 2011) and Information Retrieval systems (KARAM; MELCHIORI, 2013; BECKER; BIZER, 2009). However, to the best of our knowledge, we are the first to apply a similar improvement in a Spatial Keyword Preference query.

This chapter discusses a location-based solution that exploits the benefits of a LOD dataset for enriching the object textual description. We employ our solution at top-k Spatial Keyword Preference Query (SKPQ) (ALMEIDA; ROCHA-JUNIOR, 2016). This query accesses objects from a traditional database like OpenStreetMap. However, a LOD database like DBpedia contains objects' descriptions wider than the ones available at OpenStreetMap. In addition, DBpedia offers a semantic relationship between the stored

objects that can be explored. The contributions of this proposal are i) a novel semantic model for enhancing the SKPQ, ii) an algorithm to process the SKPQ with a LOD dataset, iii) an analysis on how the wider textual description influences the query results.

4.1 MOTIVATING SCENARIO

Several queries are processed using the Vector Space Model (VSM) to evaluate the textual relevance between query keywords and object's textual description (ALMEIDA; ROCHA-JUNIOR, 2016), (CAO et al., 2012), (CONG; JENSEN; WU, 2009). The VSM indicates that two strings are textual relevant when they share words. The top-k Spatial Keyword Preference Query (SKPQ) is a preference query that uses query keywords to describe the user preference and is processed using VSM. The SKPQ searches for spatial objects of user's interest (interesting objects) based on spatio-textual objects[1] of reference (features) in their spatial neighborhood. For example, Figure 4.1 describes a spatial area with spatial objects o (e.g. hotels) and features f (e.g. any establishment). Consider a user interested in book a hotel close to a Japanese restaurant. The user specifies the query keywords "japanese restaurant" and the spatial selection criteria (represented by the circle around the objects o). An evaluation method defines that the textual description of the object f_1 "restaurant" has textual relevance to query keywords. However, the textual description of object f_4 "japanese restaurant" is more textual relevant because it has the same words as the query keywords. Objects f_2, f_3, f_5, f_6, f_7 have no textual relevance to the query keyword, while f_5 does not satisfy the spatial selection criteria too. The SKPQ returns the object o_3 as the best hotel for the user's need, since f_4 has the greatest textual relevance among all features and satisfies the spatial selection criteria.

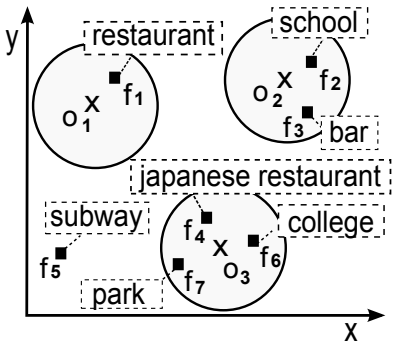

Figure 4.1 Spatial interesting objects (o) and features (f) associated with their textual descriptions

Now, suppose a SKPQ with query keywords "oriental food". Considering Figure 4.1, this query does not return any objects. Neither the word "oriental" or "food" are present in any textual description. Note that "oriental food" has semantic relevance to "japanese restaurant", but the evaluation method is not able to identify this relationship. In this example, the query fails to retrieve relevant objects when query keywords are "oriental food". So, we propose a solution using a LOD dataset to enhance the object textual

[1]Spatio-textual object is an object with spatial coordinates (e.g. latitude and longitude) and text.

description, in order to achieve better object evaluation. A wider textual description for objects f can improve the object evaluation. If object f_4 had a better textual description, the word "food" or "oriental" might appear in the textual description. In this scenario, the semantic relationship offered by the LOD dataset can be very helpful too.

4.2 PROPOSED ALGORITHM

In this section, we present the proposed algorithm to process the Spatial Keyword Preference query with LOD (SKPQ-LD). This algorithm employs SPARQL to obtain the textual description for features. The traditional SKPQ uses a Spatial Inverted Index (S2I) to index a text file with all textual descriptions needed. Before presenting the algorithm, we describe how SPARQL is used to obtain the textual description. The S2I structure is explained in Section 2.5.1.

Once an object f is found in spatial vicinity of one object o ($dist(o, f) \leq r$), its abstract is accessed using SPARQL. This abstract obtained from DBpedia is the textual description $f.D$. The textual score of f is computed using the cosine similarity function between the query keywords and the abstract $f.D$. Finally, the textual score of f is attributed to o if the score of f is higher than the current score of o ($o.score$), ($\theta(f.D, Q.D) > o.score$). The k objects with the highest scores are maintained in a heap H of k size. After computing the score of all objects o, the heap contains the k best interesting objects. Summarizing, the first step to process SKPQ requires finding all features in the spatial vicinity of each interesting object. Then, compute the textual relevance of each feature (f) to the query keywords, in order to compute the score of each object o. These steps are repeated for all interesting objects to obtain the k best ones.

In traditional SKPQ, the textual description of a feature is obtained from S2I. Given a spatial location and one term (keyword), the S2I returns one list with all features that satisfy both the textual relevance and spatial selection criteria. In this proposal, we query data from the LOD cloud with two objectives: 1) to find features that satisfy the spatial selection criteria, and 2) to obtain their textual description. Listing 4.1 describes the SPARQL query used to achieve the first objective. While the SPARQL query used to accomplish the second objective is described in Listing 4.2. More specifically, we access the DBpedia endpoint to read the abstract and comment properties values of each feature obtained. *referenceObjectURI* in Listing 4.2 is the URI to a feature.

```
SELECT DISTINCT ?resource WHERE {
        ?objectURI geo:geometry ?sourcegeo.
        ?resource geo:geometry ?location ;
        rdfs:label ?label .
FILTER( bif:st_intersects( ?location, ?sourcegeo, 0.2 ) ) . }
```

Listing 4.1 SPARQL query to find features that satisfies the spatial selection criteria

```
SELECT DISTINCT * WHERE {
        ?referenceObjectURI dbo:abstract ?abstract;
        rdfs:comment ?comment.
FILTER( lang( ?abstract)="en"&&lang(?comment)="en") }
```

Listing 4.2 SPARQL query to obtain textual description for one feature

4.2.1 The Algorithm

In traditional SKPQ, the textual description of features is previously indexed using S2I. The indexing process has a high computational cost but enables the query processing in an optimized way. Instead of computing the textual score of every feature that satisfies the spatial selection criteria (lines 5-9 of Algorithm 1), the S2I provides an iterator that accesses only the features with textual relevance and that satisfy the spatial selection criteria. The S2I avoids the score calculation of features that are in the spatial vicinity of an object of interest but has no textual relevance to the query keywords.

Algorithm 1 presents the algorithm to process the SKPQ-LD. It receives as input the SKPQ $Q = \{Q.D, Q.r, Q.k\}$, where $Q.D$ is the query keywords, $Q.r$ is the radius that defines the spatial selection criteria, and $Q.k$ is the number of expected results. The algorithm computes the score of each object $o \in O$ (lines 2-17). Initially, the score of o is zero (line 3). Then, an iterator (line 4) is employed to access all features f in the spatial vicinity of o. The textual description of each feature f is accessed (line 6), and the textual relevance between this description and the query keywords is computed (line 7) using cosine similarity. In this work, we use cosine similarity because we want the term frequency to be determinant over the document length (ZOBEL; MOFFAT, 2006). The $getAbstract(iterator.next())$ (line 6) process the SPARQL query described in Listing 4.2 to obtain the objects' textual description. After computing the score of the feature f, the function $updateScore(o, f.\theta)$ updates the score of o if the feature's textual score $f.\theta$ is higher than the current score of o (line 8).

An object o is added into H only if H has less than k objects or if the score of o is higher than the lowest score among the objects currently stored in H ($o.score > H.peekMin().score$). If the size of H is larger than k, the object with the smallest score in H is removed (lines 10-15). The algorithm returns the k objects o with the highest scores stored in H (line 17).

Algorithm 1: Processing SKPQ-LD

Input: $Q = (Q.D, Q.r, Q.k)$
Output: Heap that maintains the k best interesting objects.

```
1  H ← ∅ //Heap that maintains the k best interesting objects.
2  for each o ∈ O do
3  |  o.score ← 0
4  |  iterator ← findObjectF(objectO).iterator()
5  |  while iterator.hasNext() do
6  |  |  text ← getAbstract(iterator.next())
7  |  |  f.θ ← cosineSimilarity(text, Q.D)
8  |  |  updateScore(o, f.θ)
9  |  end
10 |  if |H| < k OR o.score > H.peekMin().score then
11 |  |  H.add(p)
12 |  |  if |H| > k then
13 |  |  |  H.removeMin()
14 |  |  end
15 |  end
16 end
17 return H
```

```
SELECT * WHERE {
        ?var rdfs:label "OSMlabel" .
        ?var geo:lat ?lat.
        ?var geo:long ?lon. }
```

Listing 4.3 SPARQL query to obtain the interesting objects to process SKPQ-LD

As shown above, the algorithm to process the SKPQ computes the score of each object $o \in O$ calculating the textual relevance between $Q.D$ and each $f' \in F'$, where F' is a subset of F ($F' \subseteq F$) that contains the feature f' that satisfies the spatial selection criteria. Hence, the complexity of the algorithm is $O(|O| \cdot |F'|)$.

4.3 RELATED WORK

Several studies employ LOD datasets to improve textual descriptions of interesting objects. Hegde et al. (HEGDE et al., 2011) describe an augmented reality browser that uses LOD to enhance the description of objects, offering a better recommendation. The objects were represented by a semantic relationship between them and several spatial data repositories such as Wikipedia[2] and YouTube. Using Natural Language Processing techniques, the user's profile is semantically related to a point of interest (POI). Then the personalized set of POIs is delivered to the user. Similarly, Karam and Melchiori (KARAM; MELCHIORI, 2013) present a way to improve POIs description using LOD.

[2]https://www.wikipedia.org/

They developed the M-PREGeD, a conceptual framework aiming to improve the accuracy of spatial data from different LOD sources. In M-PREGeD, voluntary users can generate or update POIs descriptions in order to enhance it. Aiming the same goal, we use DBpedia to enhance the textual description of features. However, we do not make use of voluntary users to help the process because we aim for an automatic enhancement approach.

The popularization of GPS (Global Positioning System) enabled devices increases significantly the volume of spatial data produced in the last years. This phenomenon stimulates new systems making use of spatial data associated with LOD. Fernández-Tobías et al. (FERNÁNDEZ-TOBÍAS et al., 2011) made use of LOD and spatial data to recommend musicians related to the architecture around the user's spatial position. Likewise our approach, they used LOD to obtain data about a spatial area (ex: architecture in Rome) but they did not make use of any spatial information (ex: latitude or longitude) in their recommendation. We use spatial information to select objects that satisfy the user's information need. Equally important, Becker and Bizer (BECKER; BIZER, 2009) presented a location-aware semantic web client for mobile devices, named DBpedia Mobile. The web client uses the current GPS position to render a map where the user can explore information about his surroundings with linked data. This information is obtained by navigating along with data links into other data repositories. In our work, we use the semantic representation of spatial objects available at DBpedia to measure the similarities between the user's keywords and the feature.

Accordingly Braun et al. (BRAUN; SCHERP; STAAB, 2010), simple text description hinders the extraction of relations between objects. In order to mitigate this problem, Braun et al. propose a semantic representation of objects using LOD. They created a collaborative spatial database compounded by POIs. In this database, users can define the ontology category of each POI. To improve the POI quality, a revision engine based on data mining techniques is provided. The revision engine identifies duplicate POIs with similar annotations or slightly varying locations for the same spatial location. In a similar fashion, Nikolaou et al. (NIKOLAOU et al., 2013) present a tool to explore linked spatial data as well as create and collaboratively edit thematic maps. Despite this tool does not provide a revision engine to improve POI quality, it provides an exploration of linked spatial data that span across multiple SPARQL endpoints. By querying these endpoints, the user is able to create his own maps and share these maps with others. The linked spatial data is explored by a tool that builds a class hierarchy and discovers the spatial extent of available information. In our work, we do not use GeoSPARQL (BATTLE; KOLAS, 2011) but we use a SPARQL extension to explore the spatial vicinity of each interesting object.

Meta-Knowledge is another approach employed to enrich the textual description. Meta-Knowledge refers to include metadata at textual corpus using an annotation scheme. For example, a news text about an event can include metadata like the modality, subjectivity, source, polarity, and specificity of the event (THOMPSON et al., 2017). This approach enriches the metadata instead of the data describing the object. In this work, we aim to enrich the data describing a spatial object. In a like manner, query expansion has long been suggested for dealing with the issue of word mismatch in information retrieval.

Accordingly to (XU; CROFT, 2017), there is a number of query expansion approaches. The main approaches are to analyze the query description to discover word relationship (global techniques) and analyze the objects retrieved by the query localization (local feedback).

With this in mind, Karpathiotaki et al. (KARPATHIOTAKI et al., 2014) introduce the Prod-Trees platform, a semantically enabled search engine for earth observation products (ex: products derived from aerial or satellite imagery). The platform has a web interface that allows users to submit free text queries. A query analyzer uses Linked Data to display different interpretations for the inserted query. The user selects the interpretation she wants, then the backend service generates queries and sends them to a catalog service. When the catalog service is ready, the results are sent to the user. In our approach, the user is able to submit free text queries as well as in the Prod-Trees platform, but we do not use a query analyzer to expand the query.

4.4 SUMMARY

This chapter presented the preliminary proposal named Spatial Keyword Preference Query with Linked Open Data (SKPQ-LD). This approach employs LOD to increase the number of words describing an interesting object to improve it. In addition, this chapter presents the proposal in details along with the algorithm to process the query using LOD. Finally, it describes the related work, comparing them with the preliminary proposal.

Chapter 5

PRELIMINARY PROPOSAL EVALUATION

In this chapter, we present our methodologies and the results obtained during the preliminary proposal evaluation. In addition, we discuss the dataset and the methodologies employed to analyze the proposed algorithm. The experiments were performed in two ways, each with a unique methodology. In the first experiment, the users' ratings from Google Maps were extracted to evaluate the queries result. In the second experiment, the users' ratings were extracted from TripAdvisor[1].

5.0.1 Datasets

In this work, we used three datasets to process the SKPQ. The OpenStreetMap[2] dataset was used to process SKPQ and, DBpedia and LinkedGeoData were used to process SKPQ-LD. Additionally, two publicly available datasets were used to evaluate the obtained query results:: the Google Maps dataset and the OpinRank dataset.

Extracts are pieces of OpenStreetMap data pruned at the region of individual continents, countries, or metropolitan areas. Mapzen[3] maintains updated extracts for many cities. In this work, we used Mapzen to obtain OpenStreetMap data from Dubai. We process this dataset to extract only spatio-textual objects. The set of objects of interest O is composed by spatial objects whose the category in the OpenStreetMap is hotel, while the set of features F is composed by the other spatio-textual objects. Table 5.0.1 presents some characteristics of the dataset obtained at Mapzen: the number of objects of interest $|O|$, the number of features $|F|$, the number of unique terms in the dataset and the total number of terms.

LinkedGeoData uses the information collected by the OpenStreetMap project and makes it available as an RDF knowledge base according to the Linked Data principles. To process SKPQ-LD we used SPARQL at LinkedGeoData to obtain a set of objects O equivalent to the one obtained from Mapzen, as illustrated by Listing 4.3. This SPARQL

[1] https://www.tripadvisor.com.br/
[2] http://www.osm.org
[3] https://mapzen.com/data/metro-extracts/

Dataset	$\|O\|$	$\|F\|$	No. of unique terms	Total number of terms
Dubai	162	2243	1906	12256

Table 5.1 Characteristics of the data obtained at Mapzen

query returns a list of objects with the same name as the one stored at Mapzen, but different spatial coordinates (i.e. there are several places called "McDonald's" in Dubai, but at different spatial coordinates). Then, we selected only the object with the same name and the same spatial coordinate as the one selected as o object at Mapzen. Additionally, we used the LinkedGeoData endpoint to access feature's textual description. The textual description obtained from LinkedGeoData is composed by *rdf:type* and *rdfs:label* predicates.

In order to enrich the object's textual description from LinkedGeoData, we used the data obtained from DBpedia. The DBpedia project has derived its data corpus from the Wikipedia encyclopedia, a large collaborative encyclopedia. When a feature has the same *rdfs:label* in DBpedia and in LinkedGeoData, we concatenate the text obtained in both datasets. The textual description $f.D$ obtained from DBpedia is composed by *rdfs:comment* and *dbo:abstract* predicates. As an example, the Hotel Danieli from Venice is described as "(tourism) (hotel) Danieli" in OpenStreetMap. While in DBpedia, the same hotel is described as "Hotel Danieli, formerly Palazzo Dandolo, is a five-star palatial hotel in Venice, Italy. (..)"[4]. The hotel description in DBpedia is much wider than the OpenStreetMap description, with 58 more words.

Both DBpedia and LinkedGeoData have public access. We accessed the data from their respective endpoints, storing the obtained data in a local repository. When the query searches for the textual description of one object, it first searches in the local repository. If the search fails, it looks for the information in the endpoints.

5.0.1.1 Dataset for Experiment 1 Besides the datasets used to process the SKPQ and SKPQ-LD, we used the Google Maps dataset and OpinRank dataset to evaluate the queries. The Google Maps dataset was accessed through the Google Places API. This dataset contains objects of interest that are updated frequently through owner-verified listings and user-moderated contributions. We extract from Google Maps the users' ratings to the hotels retrieved by the SKPQ and SKPQ-LD. These users' ratings are used to evaluate both SKPQ and SKPQ-LD.

5.0.1.2 Dataset for Experiment 2 The OpinRank dataset (GANESAN; ZHAI, 2011) contains hotel reviews and aspect ratings. There are 5 aspects ratings related to hotels: *cleanliness*, *value*, *service*, *location* and *room*. The aspect ratings values are on a scale of 1-5. Ganesan and Zhai (GANESAN; ZHAI, 2011) manually created textual queries related to each aspect rating. These queries were based on real queries made by users in popular search engines, so they reflect a natural user query. For example, the query "great location" is related to the aspect rating *location*. Given the query, the

[4]Full description can be accessed at http://dbpedia.org/page/Hotel_Danieli

Hotel name	Aspect Rating Value
Hatta Fort Hotel	4.107
Al Manzil Hotel	4.341
Park Hyatt	4.342

Table 5.2 Example of information available in OpinRank dataset related to the query "great location"

dataset lists the aspect rating value of each hotel as described in Table 5.2. The rating values are given by users from TripAdvisor when evaluating the hotels they have visited. In essence, the OpinRank dataset contains five hotels aspects, each aspect is related to five user queries and one aspect rating value for each hotel as described in Table 5.2.

5.0.2 Methodology

The DBpedia and LinkedGeoData were accessed through the local repository, or by the Snorql endpoint, as explained in Subsection 5.0.1. All experiments were executed in the same computer with an Intel Processor of 1.8 GHz (model i3-3217U) and 8 GB of RAM memory. For processing the SKPQ we made use of OpenStreetMap dataset, while for SKPQ-LD we used DBpedia dataset merged with OpenStreetMap dataset using SPARQL queries as discussed in Section 4.2.

The experiments were employed with two methodologies to evaluate the SKPQ-LD: using ratings obtained from Google Places API, and relevance judgments obtained from TripAdvisor. In Experiment 1, we apply the first methodology, where SKPQ and SKPQ-LD were executed twenty times using one unique query keyword each time. Half of the keywords are the most frequent terms in the dataset, the other half were randomly obtained. The query results were evaluated using NDCG. The list of frequent terms was obtained from S2I[5] and random queries keywords were obtained without repetition from a set of 1906 terms extracted from the OpenStreetMap dataset. "chili" and "sunset" are examples of random keywords used in this work. We used the object rate obtained from Google Places API to determine the ideal ranking.

In Experiment 2, we apply the second methodology, where SKPQ and SKPQ-LD were executed using query keywords described in the OpinRank dataset. This dataset contains full reviews of hotels collected from Tripadvisor and their corresponding aspect ratings as described in Subsection 5.0.1. We use the queries related to each aspect as query keywords and evaluate the query result obtained by SKPQ and SKPQ-LD. We ordered the query result by the aspect rating value of each hotel to determine the ideal ranking.

5.0.3 Metrics

The metrics employed in all experiments were Discount Cumulative Gain (DCG), Normalized Discount Cumulative Gain (NDCG) and Mean Average Precision (MAP). These

[5]Implementation available at XXL Library

metrics are also used in the referred related works (SONG et al., 2016; SEO et al., 2018; WANG et al., 2015). Higher values indicate better performance under these metrics.

The NDCG is widely used in Information Retrieval, measuring the quality of the ranking produced by a system (BALTRUNAS; MAKCINSKAS; RICCI, 2010; JÄRVELIN; KEKÄLÄINEN, 2002). It is particularly suitable for search applications since it accounts for multilevel relevance. The NDCG corresponds to the value of DCG divided by IDCG, defined in Equation 5.3. Since the top-k items are presented in a rank, then the Discounted Cumulative Gain (DCG) and ideal DCG (IDCG) are calculated based on Equation 5.1 and 5.2, respectively. We denote top-k items by $O_k = \{o_1, o_2, ..., o_k\}$, where the items are ranked by the SKPQ and SKPQ-LD; and we denote rel_i as the relevance value of the item at position i. DCG@k is defined as

$$DCG@k = \sum_{i=1}^{|O_k|} \frac{rel_i}{log_2(i+1)} \tag{5.1}$$

The IDCG is the maximum value of DCG. It is calculated as

$$IDCG = max(DCG@k) \tag{5.2}$$

NDCG@k is calculated as

$$NDCG@k = \frac{DCG@k}{IDCG} \tag{5.3}$$

5.0.4 Experiment 1: Evaluating Query Results

To understand the ranking quality of both SKPQ and SKPQ-LD, we compared the NDCG values obtained when using random keywords and frequent keywords. Figure 5.1 reports the arithmetic mean of NDCG@k (k=5, 10, 15, 20) that are generated by the queries with different keywords. The arithmetic mean values are reported on the vertical axis. Figures 5.0(a) and 5.0(b) illustrate that SKPQ-LD improves the ranking quality when using random keywords, otherwise the quality is roughly the same.

It is noticeable that we obtain better results with SKPQ using frequent keywords. Since the keyword is present in many objects, there is no problem to SKPQ identify the object that has textual relevance to the query keyword. In this scenario, the objects in SKPQ have a small textual description, but they have a high probability to match with the query keyword. In addition, the SKPQ access more objects because OpenStreetMap offers a larger dataset. Therefore, SKPQ counts on a good enough textual description, and a larger amount of objects, factors that lead to a better evaluation result. Nevertheless, the SKPQ-LD obtained results nearly as good as SKPQ, with a difference of only 0.1 between the NDCG values.

Figures 5.0(c) and 5.0(d) illustrate the NDCG values obtained when varying the number of query keywords. The results depicted in this Figure use a fixed k value of 5. The experiment illustrated in Figure 5.0(c) used the 10 most frequent terms in the dataset as query keywords. To build query keywords with 2 terms or more, we combined these terms with each other without repetition.

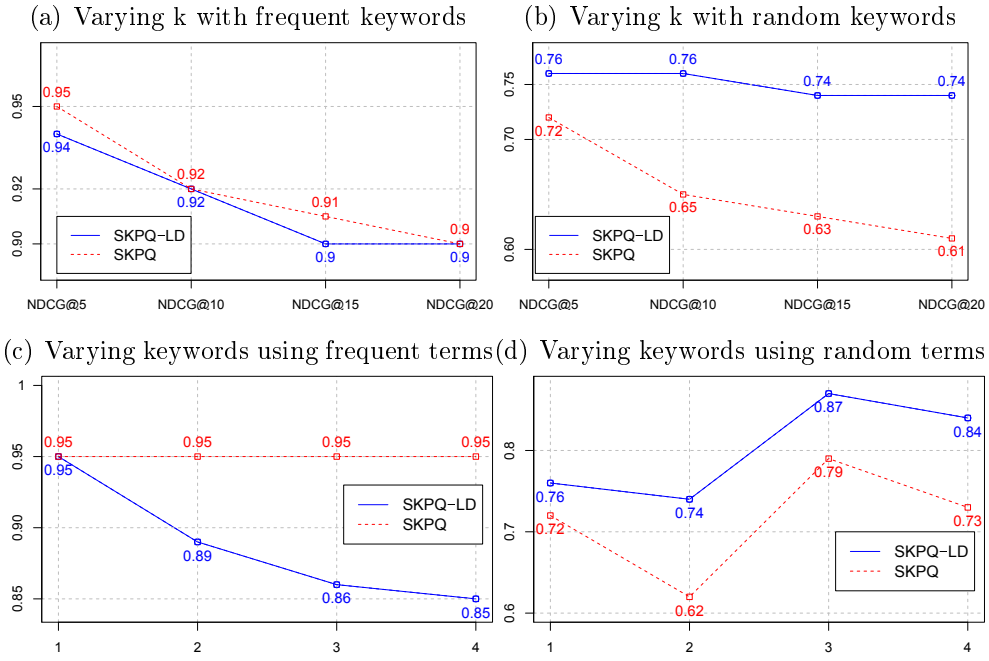

Figure 5.1 Results obtained by SKPQ and SKPQ-LD varying the keywords and the query result size (k)

As it can be seen in Figure 5.0(c), even after adding three more keywords, the results obtained in SKPQ does not change. On the other hand, SKPQ-LD is more influenced by the increase in the number of query keywords. As observed in Figure 5.1, the SKPQ presents better outcomes with frequent keywords while SKPQ-LD is better with random keywords. However, the distance between NDCG values obtained by SKPQ-LD in Figure 5.0(c) slowly decreases as the number of keywords grows. In addition, we noticed that the SKPQ results had few, or none, changes when the number of keywords was increased. For example, the query result for the keywords "parking cafe" was equal to the query results obtained with "bank parking cafe" and "parking supermarket cafe bank". The textual score of each object presented had changed, but there was no difference on the rank order, resulting in similar NDCG values. The SKPQ lacks a result variability because of the poor textual description of its objects. SKPQ-LD obtained lower NDCG values but did present different results to each query keyword.

As a baseline, the SKPQ query results are compared against the top-k Range Query (RQ) (CAO et al., 2012) results. We employ our approach to enrich the textual description of objects accessed by RQ and evaluate the results obtained. Given a spatial area and the query keyword, the RQ returns k objects in the given area that are textual relevant to the query keyword. All RQ used the same query keywords as SKPQ and a random query location in Dubai. The radius of 200 m from the selected query location defines the spatial neighborhood.

It can be seen in Figure 5.2 that our approach improved RQ result set when using frequent keywords instead of random keywords. The RQ looks for all k objects in a small

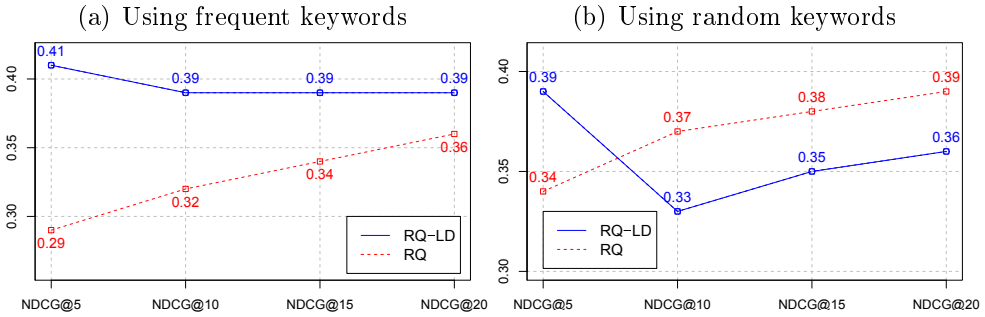

Figure 5.2 Results obtained with RQ and RQ-LD

spatial area (radius = 200 m) while SKPQ looks for objects in the neighborhood of many objects of interest. Each object neighborhood has the same size of all the spatial area visited by RQ (200 m). This contrast results in a more challenging effort to build a quality rank for the given area because there are fewer objects to verify. This can be verified observing the much lower NDCG values obtained with RQ. While SKPQ obtained 0.61 in its worst case, RQ obtained 0.41 as its best case. The amount of objects to verify is the main reason for the lower NDCGs values depicted in Figure 5.2 than the ones in Figure 5.1.

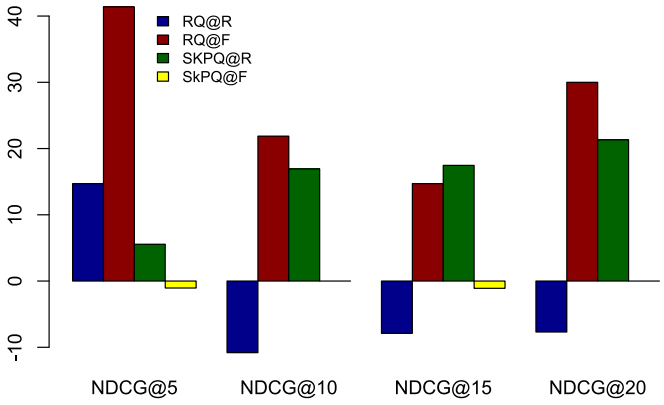

Figure 5.3 Relative NDCG improvements

Figure 5.3 illustrates the relative NDCG improvement (as described in (SONG et al., 2016)) of the proposed approach e_{pro} over respective baseline model e_{other}, further measured as

$$(e_{pro} - e_{other})/e_{other} \times 100 \tag{5.4}$$

Figure 5.3 reports the relative NDCG improvement values on the vertical axis. The proposed approach demonstrated different degrees of improvement in different scenarios. It improved SKPQ relative NDCG in 20% when using random keywords (SKPQ@R - NDCG@20) and 40% when RQ used frequent keywords (RQ@F - NDCG@5).

Using the users' ratings obtained from Google Maps, we evaluate if our approach improves the query result. Using random keywords, the hotels presented as query results

on SKPQ-LD are more popular among the users than the ones presented by the SKPQ. Using frequent keywords, the query result quality on SKPQ-LD is very similar to the one obtained by the SKPQ. Therefore, our approach does not impose a high penalty over the quality of the query result.

5.0.5 Experiment 2: Evaluating feature selection

In Experiment 2, we used the queries in OpinRank to evaluate the feature selection in SKPQ and SKPQ-LD. Since the OpinRank dataset contains only hotel reviews, we restrict our feature dataset to hotels. All hotels used in this experiment are located in Dubai.

Given the query keywords, the SKPQ returns a list of objects of interest whose are near to features relevant to the given query keywords. We desire that SKPQ returns objects whose features have a high aspect rating value. This way, the SKPQ would be selecting good features according to users of TripAdvisor. If there is no relevant feature near an interesting object, the SKPQ query result is empty.

The OpinRank dataset offers 5 textual queries for each aspect rating (total of 25 queries). These textual queries were used as query keywords in SKPQ. However, SKPQ did not find any feature whose textual description was relevant to the query keywords. The description used in SKPQ was too short and could not describe the feature as needed. Notwithstanding, the SKPQ-LD was able to find textual relevant features. From 25 queries, SKPQ-LD was able to find relevant features in 15 (equals to 60% of all executed queries). The features were retrieved with different degrees of textual relevance. Considering $k = 5$ and 25 as the number of executed queries, the MAP score obtained was 0.46.

Between the 15 relevant query results obtained by SKPQ-LD, we could extract the aspect rating value of few features. Many times, the hotel name in OpinRank dataset was not found in DBPedia or OpenStreetMap. Hence, when SKPQ or SKPQ-LD returns a hotel name that does not appear in the OpinRank dataset we can not retrieve its aspect rating value.

We show examples of textual queries that we could extract rating values, and those we could not, to illustrate this scenario. The queries “nice staff” and “good value” are examples of queries that did not return any relevant objects to the user. The objects textual description in SKPQ and SKPQ-LD was not able to describe these aspects of the hotels. However, the queries “great location”, “clean place” and “cozy rooms” returned objects when using SKPQ-LD. Figure 5.4 reports the NDCG values of the query results obtained with these query keywords.

With the enhancing of objects‘ textual description, SKPQ-LD was able to select more objects that satisfy the user need than SKPQ. Accordingly to the obtained NDCG values in Figure 5.4, SKPQ-LD selected features of good quality. Since the query results have high aspect rating values, we can assume that SKPQ-LD was able to find good objects to the user. For the query “clean place” for example, SKPQ-LD was able to find features that are evaluated by real users as a clean hotel.

The OpinRank dataset contains other queries created by the combination of the

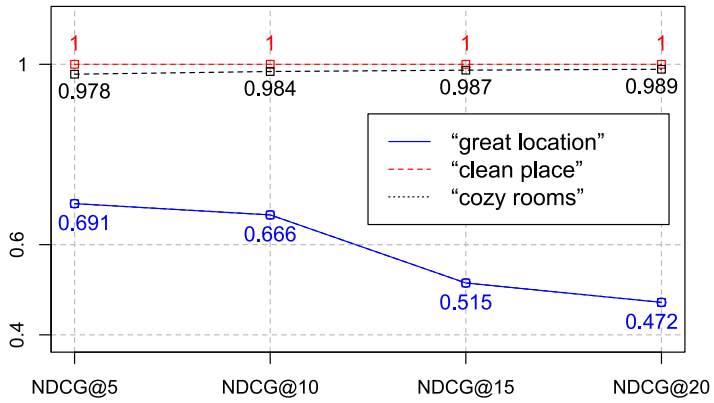

Figure 5.4 SKPQ-LD evaluation using OpinRank

queries illustrated in Figure 5.4 plus the queries "nice staff" and "good value". Nevertheless, the combination of these queries lead to results very similar to the ones at Figure 5.4. In this experiment, the SKPQ-LD demonstrated that the textual description improvement enhances the query capabilities, enabling it to find more objects. Without the textual description improvement, the SKPQ was unable to find any relevant objects to the presented queries.

5.1 DISCUSSION

In this section, we discuss how the textual description enrichment affects the results obtained in ours two experiments and the limitations of our approach. We ran the SKPQ varying query keywords and extend our analysis to understand the difference between textual descriptions from DBpedia and OpenStreetMap. Table 5.3 is an experiment result using hotels from Venice as objects of interest (O) and "church" as a query keyword. The first column of Table 5.3 presents the interesting object textual description, the second column has the interesting object score using traditional SKPQ, and the third column presents the interesting object score using SKPQ-LD. In order to find features that satisfy the spatial selection criteria using LOD, the *geo:geometry* property has to exist in LOD object. For this reason, the objects of interest "Palazzo Ferro Fini" and "Splendid Venice" has no score.

Some cities (e.g. Venice) has few spatial objects of interest represented at DBpedia. This database contains only 5 hotels in Venice against 488 registered in OpenStreetMap. Despite the great number of objects, the textual description in OpenStreetMap is poor. While a typical textual description in DBpedia has around 60 terms, the textual description for the same object has only 2 terms in OpenStreetMap. The poor textual description leads the SKPQ to misjudge the evaluation of some objects of interest, as can be seen in Table 5.3. Given the query keyword "church", objects "Hotel Cipriani" and "Hotel Danieli" have features in their spatial neighborhood that are textual relevant to the query keyword, but traditional SKPQ fails to identify them because of poor textual description. SKPQ-LD did find these objects and was able to evaluate "Hotel Cipriani" and "Hotel Danieli". For the same reason, SKPQ did not find relevant objects in the

interesting object o	**SKPQ**	**SKPQ-LD**
Hotel Cipriani	0	0.1632
Hotel Danieli	0	0.2789
Grand Hotel des Bains	0	0
Palazzo Ferro Fini	0	no *geo:geometry* property
Splendid Venice	0	no *geo:geometry* property

Table 5.3 Score of Object o in Traditional SKPQ Compared With the Score Generated by SKPQ-LD, using hotels from Venice

Experiment 2 described in Subsection 5.0.5.

interesting object o	**SKPQ**	**SKPQ-LD**
San Michel Hotel	0.5773	0.25969
Hotel Transamérica	0	0
Hotel Itamarati	0	0.2903
Hotel Braston	0	0.2596
Pousada dos Franceses	0	0.2688

Table 5.4 Score of Object o in Traditional SKPQ Compared With the Score Generated by SKPQ-LD, using hotels from São Paulo

In order to check whether the problem persists or not, we try hotels in another city. The experiment results using hotels from São Paulo as O objects and "church" as a query keyword was presented in Table 5.4. The column names in Table 5.4 have the same meaning as the column names in Table 5.3. This time we have no problem to find the *geo:geometry* property but SKPQ still has problems to evaluate objects. SKPQ still returns more objects with score zero than SKPQ-LD. These results endorse the improvement obtained by our approach when using random query keywords since "church" is a random query keyword.

As illustrated in Table 5.4, the SKPQ score of the object "San Michel Hotel" is higher than its SKPQ-LD score. When the query keyword has only one term, the textual score takes into account only the length of the document (number of terms) and the term impact. Using traditional SKPQ, we expect a higher object o score than the one computed by SKPQ-LD. The score in traditional SKPQ is higher than SKPQ-LD because the document length is shorter, therefore the term impact in this document is more evident.

5.1.1 Limitations and Points of Improvements

Despite the obtained results look promising, our approach has some limitations. First, although the LOD cloud increases every day, textual descriptions may not always be available with expected quality. This may eventually penalize the query results when using LOD. For instance, the hotel "Splendid Venice" (presented at Section 5.1) does not have the *geo:geometry* property hindering the textual description access by spatial queries.

Zarrinkalam and Kahani (ZARRINKALAM; KAHANI, 2012) describe an enrichment approach using LOD to improve the textual description of articles citations. Accordingly to him, "the Linked Data driven enrichment process has improved the quality of recommendations but it isn't as much as expected" because of "data sources that publish bibliographic information on the LOD cloud, do not yet provide adequately rich and high-quality data, compared to what these data sources provide on the web of documents".

We face the same problem with spatial information on LOD objects. LinkedGeoData has a higher amount of objects registered than DBpedia. But the textual description of objects in LinkedGeoData is poor as the ones in OpenStreetMap. In addition, a lot of less popular objects are not registered on DBpedia yet or are not well documented. Many objects do not have the *geo:geometry* property too. As a consequence, the textual description of some objects can not be enriched. For this reason, the results obtained by our approach is lower than the ones obtained by the traditional SKPQ when using frequent keywords in Experiment 1. Since the term used as the keyword is frequent in the OpenStreetMap dataset, there is no need for textual description enrichment. If we are looking for objects described as "restaurant" and all restaurants are described in the dataset, there is no need for a more detailed description. The SKPQ performs better in this context because its objects have the description needed and it has access to more objects, so it can search for more restaurants that satisfy the user need.

The world of Linked Data poses many challenges, as described in (GRACIA et al., 2012) and (BIZER et al., 2012). One meaningful challenge is the data integration in the complex and schema-less Semantic Web. However, with the fast growth of the LOD cloud, the semantic annotation becomes more popular and the datasets will provide more quality data. The proposed approach will be even more effective when more high quality data becomes more present in the web of data.

(reler ate o final)In addition, our approach relies on a distance parameter which is predefined. This input could leave out important objects of interest. This is a point of improvement that we should look at in future works. A possible solution is to personalize our approach since the user preference determines the user query in SKPQ. As a point of improvement, his profile can be considered to query expansion so that new points of interests are considered.

Despite the obtained results look promising, our approach has some limitations. Given a query Q ($Q = \{Q.D, Q.\psi, Q.k\}$), it is difficult to indicate the ideal rank to the given query. To the best of our knowledge, there is no spatial dataset with expertise judgments that could help in this scenario. Therefore we evaluate the quality of the query result using the popularity of the objects of interest from Google Maps. Given a query result R

($R = \{o_1, o_2, ..., o_k\}$), we look for the user rating given to each object in R. This is not an ideal evaluation approach because one can obtain a query result containing objects with zero similarity to the query keywords, but the user evaluation of these objects in Google Maps can be good. This divergence leads to misleading NDCG values, indicating that an object has relevance when it does not. Even so, we believe this is the best approach to evaluate the query result, measuring the average satisfaction a user would feel from obtaining the query result R.

To avoid divergences between similarity with query keywords and user ratings, as described above, we create a punishment. When the textual description of an object has no textual relevance to the query keywords, we divide its rate by two. So, if the score of o is zero ($\tau^{rng}(o) = 0$), then $o_rate = o_rate/2$

Chapter 6

This chapter presents the research stages until now. Classes completed and publications are described, as well as, the schedule.

FINAL REMARKS

6.1 ACADEMIC LIFE

In the first two years of the Ph.D. several activities took place in order to define the research topic of the research. The activities include conferences presentations and submission of journal articles. In the early stages the focus was on the Ph.D. classes and curricular components to consolidate the theoretical background. Then, the teaching internship and the workshop presentations to the research group contributed to the communication skill development.

6.1.1 Completed Classes

Each Ph.D. class gave a unique contribution to the research. In the sequence, it is described each of these contributions.

- **MATE64 - Scientific Seminars -** Professor Christina Von Flach organized a series of seminars where several professors or students in the late stage of their research presented interesting topics in computer science. The discussion after these presentations stimulate many ideas and inspired great solutions.

- **MATD74 - Algorithms and Graphs -** This class gave an overview of algorithms complexity and techniques. Professor Tiago de Oliveira Januário presented a series of challenging computational problems which students had to solve using techniques like dynamic programming, recurrences, or greedy algorithms. In addition, a paper entitled "Robot Routing: Genetic Algorithm Applied to Travelling Salesman Problem" was written to demonstrate how to solve an NP-hard problem.

- **MATE66 - Computer Science Research Fundamentals II -** The scientific methodology is one of the most important knowledge to develop solid research. This

class was lectured by Luciano Rebouças de Oliveira who motivated the students to write scientific papers objectively and efficiently. Read and writing methods were studied based on the book "Style - Toward Clarity and Grace" by Joseph M. Williams. As a result, the students produced a paper where the teacher evaluated the students' writing skills. He simulated a journal submission process, thus all papers were evaluated using a blind review method. The paper wrote in this class was the start of the first paper published as a product of this research.

- **MATE32 - Topics on Computer Intelligence II -** Overview of machine learning algorithms focusing on automatic knowledge retrieval from datasets. Professors Ricardo Araújo Rios and Tatiane Nogueira Rios explained how to pre-process data properly and to analyze data using predictive probabilistic methods based on optimization. This class was focused on supervised learning algorithms and how to evaluate them.

- **MATE33 - Topics on Computer Intelligence III -** Professors Ricardo Araújo Rios and Tatiane Nogueira Rios taught about use clustering methods and unsupervised learning algorithms. The students were challenged with clustering problems and led to solving these problems using the main techniques available in the related literature. This class was essential to improve the skill of pattern identification on datasets.

- **MATE85 - Topics on Information Systems and Web I** - This class lectured by the advisor presented core topics on Linked Open Data and Web Semantic. The students developed projects using technologies related to this topic, like Jena and Protégé. This class did a substantial contribution to the research and software development.

- **MATA31 - Oriented Research -** This curricular component is used by the advisor to evaluate the research progress. Periodical feedbacks were given to the advisor who led the research to the correct path with suggestions and contributions.

- **MATA32 - Oriented Teaching Internship -** The advisor offered to the Ph.D. student the experience of teaching to undergraduate students. Additionally, the student gave support for lectures preparation and assisted the class on projects.

- **Classes Dispensed -** Some classes were valuable to the research but they were completed during the master degree on Programa de Pós-Graduação em Computação Aplicada (PGCA). For this reason, the following classes were dispensed by the Programa de Pós-Graduação em Ciência da Computação (PGCOMP):

  - MATE04 - Topics on Databases I
  - MATE10 - Topics on Computer Intelligence I

6.1.2 Publications and Participation in Scientific Conferences

It is important to the Ph.D. student to participate in scientific conferences and to publish the research results on relevant journals. Conferences are a channel to disclose and discuss the research, exchanging information with other researchers from related areas. Following are described the published papers and the conferences attended.

- **WebMedia 2018 - Brazilian Symposium on Multimedia and the Web -** it is the main event of the theme in Brazil and an excellent opportunity for scientific and technical exchange among students, researchers and professionals in the areas of Multimedia, Hypermedia and Web. We published the paper (ALMEIDA; DURÃO, 2018) presenting the preliminary results of our method to enhance the SKPQ accuracy using Linked Open Data.

- **J.UCS 2018 - Journal of Universal Computer Science -** this journal is run by the J.UCS consortium consisting of research institutions from Austria, Germany, Guatemala, USA, and Pakistan. We published the paper (ALMEIDA; DURÃO; COSTA, 2018) detailing our approach to enhance the SKPQ and discussed all obtained results in the experimental evaluation.

- **AMCIS 2019 - Annual Americas Conference on Information Systems -** AMCIS is viewed as one of the leading conferences for presenting the broadest variety of research done by and for information technology academicians. Every year its papers and panel presentations are selected from over 700 submissions, and the AMCIS proceedings are in the permanent collections of libraries throughout the world. From a collaborative work we did a paper about exploiting Web features for relevance feedback. This paper has been accepted and it is in the publishing process.

- **WE.PGCOMP 2018 -** this conference is a curricular component too. After the second year as a Ph.D. student, once per year, the student has to present his research progress to three professors of the doctoral program and to an audience. The research progress is evaluated by the professors who ask questions and make great suggestions for the research.

- **ESWA 2019 - Expert Systems With Applications -** is a refereed international journal whose focus is on exchanging information relating to expert and intelligent systems applied in industry, government, and universities worldwide. The personalization approach we described to improve the SKPQ will be reported in a paper which will be submitted to this journal.

- **Data Mining and Knowledge Discovery 2020 -** the premier technical publication in the field, it is a resource collecting relevant common methods and techniques and a forum for unifying the diverse constituent research communities. We plan to submit a paper to this journal in 2020 describing new techniques and experimental evaluations.

6.2 SCHEDULE

Until now we have proposed two techniques to improve spatial keyword queries accuracy. There were several experimental evaluations demonstrating query improvement. However, there is room to further improve our work. Under those circumstances, Figure 6.1 describes the activities of this research. Activities A01 to A10 was developed before the qualification exam, while A10 to A17 will be developed before the thesis defense.

- **Activity A01 - Academic Classes -** describes the time consumed to complete all classes demanded by the doctoral program. During this time the theoretical background was consolidated. All classes have been described in Section 6.1.1
- **Activity A02 - Literature Review -** studies related to the research topic definition. Many papers were read and drafts were written during this process.
- **Activity A03 - Topic discussion -** meetings with the advisor to discuss research topics and possible solutions to these topics.
- **Activity A04 - Workshops -** meetings involving the research members of the RecSys group. Each meeting consists of members presentation about their research topics followed by practices exercises. There was a workshop about SPARQL queries and another about Jena and Web Semantic.
- **Activity A05 - First proposal: LOD enhancement to spatial queries -** studies related to the research topic definition, writing the proposal definition and define the strategies to solve the problem.
- **Activity A06 - Implementation of the first proposal -** development of the strategies or algorithms defined in the early stage.
- **Activity A07 - Preliminary experiments -** the first rounds of experiments on our approach based on LOD to improve spatial keyword queries.
- **Activity A08 - Evaluation of preliminary results -** analysis of the results with graphs and proper metrics.
- **Activity A09 - Submission to J.UCS 2018 -** the first publication inside the doctoral program. This activity consumed much time because it was the first related to this topic. In this paper, we talked about the benefits of using LOD to enhance textual descriptions of objects and how it improves spatial keyword queries. Several evaluation experiments were described and analyzed.
- **Activity A10 - Submission to WebMedia 2018 -** the first paper submitted to a conference during the doctoral research. Motivated by the late review response on the J.UCS article, in this paper, we described some of the experiments on the LOD enhancement. The conference was a pleasurable experience where was possible to discuss the research topic with experienced researchers.

- **Activity A11 - Extension of the first proposal: query personalization -** after the first publication, the work on the second proposal started. In this activity we worked on the personalization of spatial keyword queries. The experimental evaluation indicates a relevant improvement on these queries accuracy.

- **Activity A12 - Implementation of the proposal extension -** implementation of the strategies or algorithms defined in the early stage.

- **Activity A13 - Preliminary experiments of the proposal extension -** initial experiments on our approach to personalize spatial keyword queries.

- **Activity A14 - Submission to AMCIS 2019 -** in this activity a collaborative work was done, revising the article and attending the demands of the journal's reviewers.

- **Activity A15 - Submission to ESWA 2019 -** the second paper describing the personalization approach is submitted and we are waiting for the journal response.

- **Activity A16 - Qualification -** write qualification text and prepare the qualification presentation.

- **Activity A17 - New optimization research -** explore new methods to improve spatial keyword queries. New algorithms or hybrid solutions can be proposed in this stage.

- **Activity A18 - Implementation of the new optimization -** this activity will implement the strategies or algorithms defined in the previous activity.

- **Activity A19 - Experiment results -** after the development stage it is necessary to evaluate the algorithms or method developed.

- **Activity A20 - Experiment analysis -** the experimental results are analyzed and compared with previous results, as well as, with the results reported in the literature.

- **Activity A21 - Preliminary results paper -** another paper will be published introducing the new approach and the results obtained so far. It is possible to submit on paper to a journal and another to a conference in this stage.

- **Activity A22 - Thesis -** activity defined to write the thesis.

- **Activity A23 - Thesis defense -** designated to thesis presentation and research end.

The schedule of activities executed since the course enrollment together with the ones that will be executed after the qualification exam are presented in Figure 6.1.

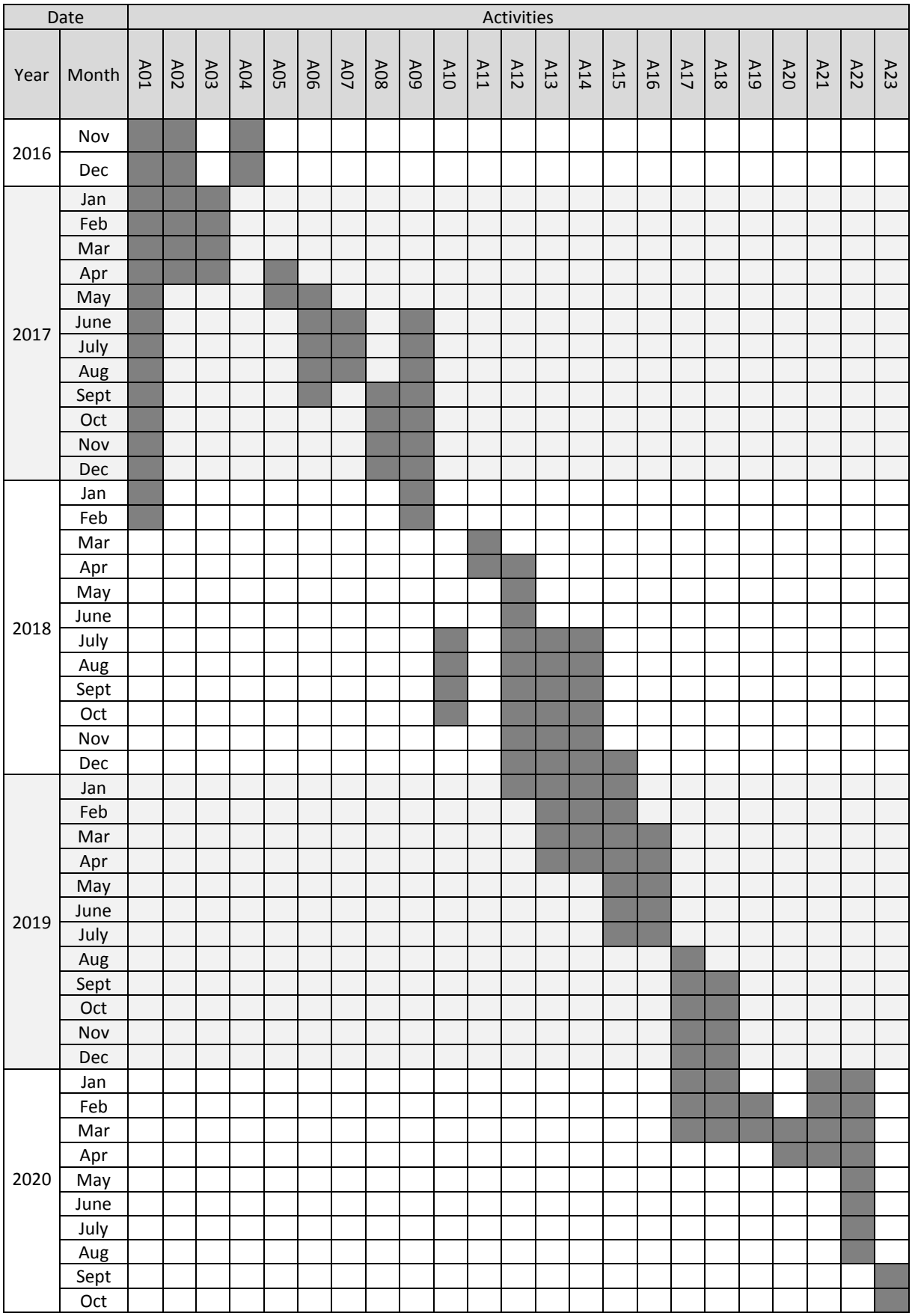

Figure 6.1 Ph.D. activities since the course enrollment until thesis defense.

6.3 SUMMARY

The aforementioned chapter described the student life academy until now. It was depicted the classes completed and how they contributed to the research. In addition, it is possible to see the timeline between the publications and the stages before them. On the other hand, the schedule illustrates the time left to the thesis defense and future works followed by each scheduled date.

BIBLIOGRAPHY

ALMEIDA, J. P. D. de; DURÃO, F. A. Improving the spatial keyword preference query with linked open data. In: SBC. *Anais Estendidos do XXIV Simpósio Brasileiro de Sistemas Multimídia e Web*. [S.l.], 2018. p. 19–24.

ALMEIDA, J. P. D. de; DURÃO, F. A.; COSTA, A. F. da. Enhancing spatial keyword preference query with linked open data. v. 24, n. 11, p. 1561–1581, nov 2018.

ALMEIDA, J. P. D. de; ROCHA-JUNIOR, J. B. Top-k spatial keyword preference query. *Journal of Information and Data Management*, p. 162, 2016.

BALTRUNAS, L.; MAKCINSKAS, T.; RICCI, F. Group recommendations with rank aggregation and collaborative filtering. In: ACM. *Proceedings of the fourth ACM conference on Recommender systems*. [S.l.], 2010. p. 119–126.

BARUFFOLO, A. R-trees for astronomical data indexing. In: *Astronomical Data Analysis Software and Systems VIII*. [S.l.: s.n.], 1999. v. 172, p. 375.

BATTLE, R.; KOLAS, D. Geosparql: enabling a geospatial semantic web. *Semantic Web Journal*, p. 355–370, 2011.

BAYER, R.; MCCREIGHT, E. Organization and maintenance of large ordered indexes. In: ACM-SIGFIDET. *Workshop on Data Description and Access*. Houston, Texas, 1970.

BECKER, C.; BIZER, C. Exploring the geospatial semantic web with dbpedia mobile. *Web Semantics: Science, Services and Agents on the World Wide Web*, Elsevier, p. 278–286, 2009.

BECKMANN, N. et al. The R*-tree: An efficient and robust access method for points and rectangles. In: *SIGMOD*. [S.l.]: ACM, 1990. p. 322–331.

BENTLEY, J. L.; FRIEDMAN, J. H. Data structures for range searching. *ACM Computing Surveys (CSUR)*, ACM, v. 11, n. 4, p. 397–409, 1979.

BERNERS-LEE, T. Design issues: Linked data (2006). *URL http://www.w3.org/DesignIssues/LinkedData.html*, 2011.

BERNERS-LEE, T.; FIELDING, R.; MASINTER, L. Rfc 3986. *Uniform Resource Identifier (URI): Generic Syntax*, InternetEngineering Task Force, 2005.

BIZER, C. et al. The meaningful use of big data: four perspectives–four challenges. *ACM SIGMOD Record*, ACM, v. 40, n. 4, p. 56–60, 2012.

BIZER, C.; HEATH, T.; BERNERS-LEE, T. Linked data-the story so far. *Semantic services, interoperability and web applications: emerging concepts*, p. 205–227, 2009.

BORZSONY, S.; KOSSMANN, D.; STOCKER, K. The skyline operator. In: IEEE. *Data Engineering, 2001. Proceedings. 17th International Conference on*. [S.l.], 2001. p. 421–430.

BRAUN, M.; SCHERP, A.; STAAB, S. Collaborative creation of semantic points of interest as linked data on the mobile phone. In: *Extended Semantic Web Conference (Demo Session)*. [S.l.]: Springer, 2010.

CAO, X. et al. Spatial keyword querying. In: *ER*. [S.l.]: Springer, 2012. p. 16–29.

CARMEL, D.; GUETA, G.; BORTNIKOV, E. *Top-k query processing with conditional skips*. [S.l.]: Google Patents, 2018. US Patent App. 15/345,277.

CHEN, L. et al. Spatial keyword query processing: an experimental evaluation. In: . [S.l.]: VLDB Endowment, 2013. p. 217–228.

CHOMICKI, J. Preference formulas in relational queries. *ACM Transactions on Database Systems (TODS)*, ACM, v. 28, n. 4, p. 427–466, 2003.

COHEN, W. W.; RAVIKUMAR, P.; FIENBERG, S. E. A comparison of string distance metrics for name-matching tasks. *Workshop on Data Cleaning and Object Consolidation*, p. 73–78, 2003.

COMER, D. Ubiquitous b-tree. *ACM Computing Surveys (CSUR)*, ACM, v. 11, n. 2, p. 121–137, 1979.

CONG, G.; JENSEN, C. S.; WU, D. Efficient retrieval of the top-k most relevant spatial web objects. In: . [S.l.]: VLDB Endowment, 2009. v. 2, n. 1, p. 337–348.

FERNÁNDEZ-TOBÍAS, I. et al. A generic semantic-based framework for cross-domain recommendation. In: ACM. *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*. [S.l.], 2011. p. 25–32.

FIELDING, R. et al. *Hypertext transfer protocol–HTTP/1.1*. [S.l.], 1999.

GANESAN, K.; ZHAI, C. Opinion-based entity ranking. *Information Retrieval*, 2011.

GRACIA, J. et al. Challenges for the multilingual web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, Elsevier, v. 11, p. 63–71, 2012.

GÜTING, R. H. An introduction to spatial database systems. *The VLDB Journal–The International Journal on Very Large Data Bases*, Springer-Verlag New York, Inc., v. 3, n. 4, 1994.

GUTTMAN, A. *R-trees: a dynamic index structure for spatial searching*. [S.l.]: ACM, 1984. v. 14.

GUTTMAN, A.; STONEBRAKER, M. Using a relational database management system for computer aided design data. *IEEE Database Eng. Bull.*, v. 5, n. 2, p. 21–28, 1982.

HARIHARAN, R. et al. Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems. In: IEEE. *Scientific and Statistical Database Management, 2007. SSBDM'07. 19th International Conference on*. [S.l.], 2007. p. 16–16.

HEGDE, V. et al. Utililising linked data for personalized recommendation of poi's. In: *International AR Standards Meeting, Barcelona, Spain*. [S.l.: s.n.], 2011.

ILYAS, I. F.; BESKALES, G.; SOLIMAN, M. A. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys (CSUR)*, ACM, v. 40, n. 4, p. 11, 2008.

JÄRVELIN, K.; KEKÄLÄINEN, J. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, ACM, v. 20, n. 4, p. 422–446, 2002.

KARAM, R.; MELCHIORI, M. Improving geo-spatial linked data with the wisdom of the crowds. In: ACM. *Proceedings of the joint EDBT/ICDT 2013 workshops*. [S.l.], 2013. p. 68–74.

KARPATHIOTAKI, M. et al. Prod-trees: semantic search for earth observation products. In: SPRINGER. *European Semantic Web Conference*. [S.l.], 2014. p. 374–378.

KELES, I. *Spatial Keyword Querying: Ranking Evaluation and Efficient Query Processing*. Tese (Doutorado) — Aalborg Universitetsforlag, 2018.

KHODAEI, A.; SHAHABI, C.; LI, C. Hybrid indexing and seamless ranking of spatial and textual features of web documents. In: SPRINGER. *Database and Expert Systems Applications*. [S.l.], 2010. p. 450–466.

KIESSLING, W. Foundations of preferences in database systems. In: VLDB ENDOWMENT. *Proceedings of the 28th international conference on Very Large Data Bases*. [S.l.], 2002. p. 311–322.

KLYNE, G.; CARROLL, J. J. Resource description framework (rdf): Concepts and abstract syntax. 2006.

LACROIX, M.; LAVENCY, P. Preferences; putting more knowledge into queries. In: *VLDB*. [S.l.: s.n.], 1987. v. 87, p. 1–4.

LI, G. et al. Reverse top-k query on uncertain preference. In: SPRINGER. *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*. [S.l.], 2018. p. 350–358.

LI, Z. et al. Ir-tree: An efficient index for geographic document search. *Knowledge and Data Engineering, IEEE Transactions on*, IEEE, v. 23, n. 4, p. 585–599, 2011.

MACKENZIE, J.; CHOUDHURY, F. M.; CULPEPPER, J. S. Efficient location-aware web search. In: ACM. *Proceedings of the 20th Australasian Document Computing Symposium*. [S.l.], 2015. p. 4.

MANNING, C. D. et al. *Introduction to information retrieval*. [S.l.]: Cambridge university press, 2008. v. 1.

MENG, X.; ZHANG, X.; ZHAO, Z. T k qs: A top-k keyword query suggestion system. In: IEEE. *2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. [S.l.], 2018. p. 1005–1008.

NIKOLAOU, C. et al. Sextant: browsing and mapping the ocean of linked geospatial data. In: SPRINGER. *Extended Semantic Web Conference*. [S.l.], 2013. p. 209–213.

PAN, J. Z. Resource description framework. In: *Handbook on ontologies*. [S.l.]: Springer, 2009. p. 71–90.

PAPADIAS, D. et al. Efficient OLAP operations in spatial data warehouses. In: *SSTD*. [S.l.]: Springer, 2001. p. 443–459.

PERRY, M.; HERRING, J. Ogc geosparql-a geographic query language for rdf data. *OGC Implementation Standard. Sept*, 2012.

RIGAUX, P.; SCHOLL, M.; VOISARD, A. *Spatial databases: with application to GIS*. [S.l.]: Morgan Kaufmann, 2001.

ROCHA-JUNIOR, J. B. *Efficient processing of preference queries in distributed and spatial databases*. Tese (Doutorado) — Norwegian University of Science and Technology, 2012.

ROCHA-JUNIOR, J. B. et al. Efficient processing of top-k spatial keyword queries. In: *SSTD*. [S.l.]: Springer, 2011. p. 205–222.

SALMINEN, A.; TOMPA, F. W. Pat expressions: an algebra for text search. *Acta Linguistica Hungarica*, v. 41, n. 1, p. 277–306, 1994.

SAMET, H. The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, ACM, v. 16, n. 2, p. 187–260, 1984.

SEO, Y.-D. et al. An enhanced aggregation method considering deviations for a group recommendation. *Expert Systems with Applications*, Elsevier, v. 93, p. 299–312, 2018.

SHANBHAG, A.; PIRK, H.; MADDEN, S. Efficient top-k query processing on massively parallel hardware. In: ACM. *Proceedings of the 2018 International Conference on Management of Data*. [S.l.], 2018. p. 1557–1570.

SONG, H. et al. Individual judgments versus consensus: Estimating query-url relevance. *ACM Transactions on the Web (TWEB)*, ACM, v. 10, n. 1, p. 3, 2016.

THOMPSON, P. et al. Enriching news events with meta-knowledge information. *Language Resources and Evaluation*, Springer, v. 51, n. 2, p. 409–438, 2017.

WANG, J. et al. Diversionary comments under blog posts. *ACM Transactions on the Web (TWEB)*, ACM, v. 9, n. 4, p. 18, 2015.

WU, D.; CONG, G.; JENSEN, C. S. A framework for efficient spatial web object retrieval. *The VLDB Journal–The International Journal on Very Large Data Bases*, Springer-Verlag New York, Inc., v. 21, n. 6, p. 797–822, 2012.

WU, D. et al. Joint top-k spatial keyword query processing. *Knowledge and Data Engineering, IEEE Transactions on*, IEEE, v. 24, n. 10, p. 1889–1903, 2012.

XU, J.; CROFT, W. B. Quary expansion using local and global document analysis. In: ACM. *Acm sigir forum*. [S.l.], 2017. v. 51, n. 2, p. 168–175.

YIU, M. L. et al. Top-k spatial preference queries. In: IEEE. *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. [S.l.], 2007. p. 1076–1085.

ZARRINKALAM, F.; KAHANI, M. A multi-criteria hybrid citation recommendation system based on linked data. In: IEEE. *Computer and Knowledge Engineering (ICCKE), 2012 2nd International eConference on*. [S.l.], 2012. p. 283–288.

ZHOU, Y. et al. Hybrid index structures for location-based web search. In: ACM. *Proceedings of the 14th ACM international conference on Information and knowledge management*. [S.l.], 2005. p. 155–162.

ZHU, S. et al. Scaling up top-K cosine similarity search. *Data & Knowledge Engineering*, Elsevier, p. 60–83, 2011.

ZOBEL, J.; MOFFAT, A. Inverted files for text search engines. In: . [S.l.]: ACM, 2006. p. 6.