National Technical University of Athens

School of Electrical and
Computer Engineering

Knowledge and Database Systems Laboratory

Geo-Spatial Knowledge in User-Generated Data: Mining, Modeling and Applications

PhD Thesis

of

Georgios Skoumas

Diploma in Electronics and Computer Engineering (TUC 2008)
MPhil in Computer Science (Cambridge 2010)

Athens, July 2015

National Technical University of Athens
School of Electrical and Computer Engineering
Knowledge and Database Systems Laboratory

Geo-Spatial Knowledge in User-Generated Data: Mining, Modeling and Applications

PhD Thesis

of

Georgios Skoumas

Diploma in Electronics and Computer Engineering (TUC 2008)
MPhil in Computer Science (Cambridge 2010)

Supervising Committee: T. Sellis
I. Vasileiou
D. Pfoser

Approved by the Examination Committee, September 2015.

...	...	...
T. Sellis	I. Vasileiou	D. Pfoser
Prof. RMIT	Prof. NTUA	Assoc. Prof. GMU
...	...	...
G. Stamou	T. Dalamangas	M. Kavouras
Assist. Prof. NTUA	Reserecher IMIS	Prof. NTUA
	...	
	D. Gunopulos	
	Prof. UoA	

Athens, September 2015

...

Georgios Skoumas
Diploma in Electronics and Computer Engineering (TUC 2008)
MPhil in Computer Science (Cambridge 2010)

ABSTRACT

With the proliferation of the Internet as the primary medium for data publishing and information exchange, we have seen an explosion in the amount of online content available on the Web. Thus, in addition to professionally-produced material being offered free on the Internet, the public has also been allowed, indeed encouraged, making its content available online to everyone. The volumes of such User-Generated Content (UGC) are already staggering and constantly growing. Our goal has to be to take advantage of this data explosion, which applied to the spatial domain translates to massively collecting and sharing knowledge to ultimately digitize the world. User-generated geospatial content is also commonly referred to as Volunteered Geographic Information (VGI). There are several forms of VGI. In the current thesis, we will work on VGI from textual and GPS data.

Subsumed under VGI, non-expert users have been providing a wealth of quantitative geospatial data online. With spatial reasoning being a basic form of human cognition, narratives expressing geospatial experiences, e.g., travel blogs, would provide an even bigger source of geospatial data. Textual narratives typically contain qualitative data in the form of objects and spatial relationships. One of the main scopes of this thesis is (*i*) to extract these relationships from user-generated texts, (*ii*) to quantify them and (*iii*) to reason about object locations based only on this qualitative data.

Moreover, with the extracted and modeled spatial relations and by employing Bayesian inference, we obtain probabilistic measures of spatial connectedness of PoIs according to the crowd. Applying this measure to the corresponding road network, we obtain an altered cost function which does not exclusively rely on distance, and enriches an actual road networks taking crowdsourced spatial relations into account. With this we aim at obtaining paths that do not only minimize distance but also lead through more popular areas using knowledge generated by users.

The last objective of this thesis, is to introduce the problem of continuous and non-continuous monitoring of nearest trajectories based on GPS data. In contrast to other similar approaches, we are interested in monitoring moving objects taking into account at each timestamp not only their current positions but their recent trajectory in a defined time window. We first describe generic baseline algorithms for this problem, which applies for any aggregate function used to compute trajectory distances between objects, and without any restrictions on the movement of the objects. Using this as a framework, we continue to derive optimized algorithm for the cases where the distance between two moving objects in a time window is determined by their maximum or minimum distance in all contained timestamps. Furthermore, we propose additional optimizations for the case that an upper bound on the velocities of the objects exists.

Key words: User generated content, spatial relations, probabilistic modeling, location estimation, popular path computation, trajectory mining, knn

ΠΕΡΙΛΗΨΗ

Με την εξάπλωση του Διαδικτύου ως το κύριο μέσο για τη δημοσίευση στοιχείων και την ανταλλαγή πληροφοριών, έχουμε δει μια έκρηξη στον όγκο του περιεχομένου που είναι διαθέσιμο στο διαδίκτυο. Έτσι, εκτός από επαγγελματικό περιεχόμενο που είναι διαθέσιμο στο διαδίκτυο, το κοινό έχει επίσης τη δυνατότητα, να καθιστά το περιεχόμενό του διαθέσιμο σε όλους. Οι όγκοι της εν λόγω πληροφορίας από τους χρήστες είναι συνεχώς αυξανόμενοι. Στόχος μας πρέπει να είναι να επωφεληθούμε από αυτή την έκρηξη των δεδομένων, η οποία εφαρμοζόμενη στο γεοχωρικό πεδίο μεταφράζεται στη μαζική τη συλλογή και την ανταλλαγή γνώσης για την ψηφιοποίηση του κόσμου. Το περιεχόμενο που δημιουργείται από χρήστες στο γεωχωρικό πεδίο επίσης, αναφέρεται και ως εθελοντική γεωγραφική πληροφορία. Η εθελοντική γεωχωρική πληροφορία μπορεί να εμπεριέχεται σε πολλύς τύπους δεδομένων. Στην παρούσα διατριβή, θα ασχοληθούμε με γεωχωρική πληροφορία από δεδομένα κειμένου και από δεδομένα πλοήγησης.

Υπό τον όρο εθελοντική γεωγραφική πληροφορία, μη εξειδικευμένοι χρήστες παρέχουν έναν πλούτο ποσοτικών γεωχωρικών δεδομένων στο διαδίκτυο. Με τη χωρική συλλογιστική να είναι μια βασική μορφή της ανθρώπινης νόησης, αφηγήσεις που εκφράζουν γεωχωρικών εμπειρίες, π.χ., ταξιδιωτικά ιστολόγια, παρέχουν μια ακόμη μεγαλύτερη πηγή γεωχωρικών δεδομένων. Οι κειμενικές αφηγήσεις συνήθως περιέχουν ποιοτικά δεδομένα με τη μορφή χωρικών αντικειμένων και χωρικών σχέσεων. Ένας από τους βασικούς στόχους της παρούσας διατριβής είναι (1) για να εξαγάγουμε τις σχέσεις αυτές από κείμενα που δημιουργούνται από χρήστες, (2) να τις ποσοτικοποιήσουμε και (3) και να βγάλουμε συμπεράσματα για τις θέσεις αντικειμένων στο χώρο βασιζόμενοι μόνο σε αυτά τα ποιοτικά δεδομένα.

Ένας επιπλέον στόχος της παρούσας διατριβής, είναι η χρησιμοποίηση των εξαγόμενων και ποσοτικά μοντελοποιημένων χωρικων σχέσεων σε συνδυασμό με πιθανοτική θεωρία, και την εφαρμογή τους για την επίλυση του προβλήματος του δημοφιλούς μονοπατιού. Συγκεκριμένα, χρησιμοποιούμε την χωρική πληροφορία που παράγεται από χρήστες και με αυτό στοχεύουμε στην απόκτηση μονοπατιών που δεν ελαχιστοποιούν μόνο την απόσταση, αλλά επίσης οδηγούν σε πιο δημοφιλείς περιοχές χρησιμοποιώντας γνώσης που παράγεται από τους ίδιους τους χρήστες.

Ο τελευταίος στόχος της παρούσας διπλωματικής εργασίας, είναι να εισάγει το πρόβλημα της παρακολούθησης πλησιέστερων τροχιών με βάση δεδομένα πλοήγησης. Μελετούμε το πρόβλημα αυτό στη συνεχή και στη μή συνεχή περίπτωση. Παρουσιάζουμε αρχικά γενικούς και βασικούς αλγόριθμους για το πρόβλημα αυτό, και στην συνέχεια προτείνουμε επιπλέον βελτιστοποιήσεις με ακριβής και προσεγγιστικούς αλγορίθμους χρησιμοποιώντας πολλών τύπων χαρακτηριστικά.

ACKNOWLEDGMENTS

First, I would like to express my gratitude to my supervisors Timos Sellis and Dieter Pfoser. They provided me with a deep perspective of crowdsourcing, data mining and knowledge extraction and guidance on many technical aspects of this thesis. Moreover, I would like to thank Yannis Vasiliou, Georgios Stamou, Thodoris Dalamangas, Marinow Kavouras and Dimitrios Gunopoulos for agreeing to serve as members of my final examination committee.

During my time in DBLAB, I had the great pleasure to collaborate with great researchers. I would like to thank all my lab members: Christodoulos, Sofia, George, Aris, Alex, Dimitris Skoutas and Dimitris Sacharidis. I truly enjoyed spending my time with them and I have learned a lot from each one of them. I'm also grateful to be lab-mate with Giorgos Giannopoulos, Giannis Liagouris and Nikos Bikakis.

Many thanks go to the DataBase group at LMU Munich for hosting me as a research intern during the winter of 2014. In particular, I would like to thank Gregor Josse, Klaus Arthur Schmid, Andreas Zufle, Matthias Renz, Mario Nascimento and Irene Ntoutsi for all your great help and amazing moments we had during my stay in Munich. I consider all of you as beloved friends and I wish we will have the opportunity to live more amazing moments in the future.

Finally, my heartfelt thanks go out to my family and friends in Greece for their constant support and encouragement.

Athens, July 2015 G. S.

Αν του θυμού σου τις στιγμές, που φαίνεται αδυσώπητη η ψυχή σου,
μπορείς ν' αφήσεις να διαβούν την πρώτη ξαναβρίσκοντας γαλήνη,
δική σου θάναι τότε η Γη, μ' όλα και μ' ό,τι πάνω της κι' αν έχει
και κάτι ακόμα πιο πολύ: Άντρας αληθινός θά΄σαι, παιδί μου
— Rudyard Kipling

To my family...

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Prologue

With the proliferation of the Internet as the primary medium for data publishing and information exchange, we have seen an explosion in the amount of online content available on the Web. Thus, in addition to professionally-produced material being offered free on the Internet, the public has also been allowed, indeed encouraged, making its content available online to everyone. The volumes of such User-Generated Content (UGC) are already staggering and constantly growing.

In this dissertation, our goal is to take advantage of this data explosion, which applied to the spatial domain translates to massively collecting and sharing knowledge to ultimately digitize the world. User-generated geospatial content is also commonly referred to as Volunteered Geographic Information (VGI). Subsumed under the term VGI, non-expert users have been providing a wealth of qualitative and qualitative geospatial data online.

With spatial reasoning being a basic form of human cognition, narratives expressing geospatial experiences, e.g., travel blogs, social media etc., would provide an even bigger source of geospatial data. Use generated texts typically contain qualitative data in the form of objects and spatial relationships. One of the main objectives of this thesis is to extract this geospatial information, model it under a probabilistic framework and utilize these models in order to solve real world problems. More specifically, we propose some new algorithms for on location estimation of unknown Points-of-interest (POIs) which is divided in three steps: i we extract spatial relationships from user-generated texts, (ii) we quantify them and (iii) we reason about object locations based only on this qualitative data.

The second objective of this dissertation is to provide new approaches for the popular path Computation problem. Moving to this direction, we employ extracted and modeled spatial relations and by Bayesian inference, we obtain probabilistic measures of spatial connectedness of PoIs according to the crowd. Applying this measure to the corresponding road network, we propose an approach that enriches an actual road networks taking crowdsourced spatial relations into account. With this we aim at obtaining paths that do not only minimize distance but also lead through more popular areas using knowledge generated by users.

Finally, the last objective of this thesis, is the analysis of UGC in the form of GPS traces. Towards this research direction, we propose some algorithms on the problems of non-continuous and continuously monitoring of nearest trajectories

based on GPS data. In contrast to other similar approaches, we are interested in monitoring moving objects taking into account at each timestamp not only their current positions but their recent trajectory in a defined time window.

1.2 Contribution

The contribution of this dissertation may be divided into four major sections. We will describe each of these sections briefly along with the individual scientific contributions per section:

1.2.1 Mining and Modeling Geospatial Data

Our first contribution includes an efficient approach to mine geospatial content, i.e., spatial relations, from user generated texts and a novel probabilistic method to model qualitative spatial relations under a quantitative probabilistic framework.

Specifically, we choose travel blogs as a potentially rich geospatial data source and we use Natural Language Processing (NLP) algorithms and tools in order to extract spatial relations. Obtaining qualitative spatial relations from text involves the detection of (*i*) spatial objects, i.e., Points-of-Interest (POIs) or toponyms, and (*ii*) spatial relationships linking the POIs. Our approach involves geoparsing, i.e., the detection of candidate phrases, and geocoding, i.e., linking parts of phrase/toponym to actual coordinate information. The proposed approach achieves notable accuracy even with noisy crowdsourced data.

Continuing, we model the extracted spatial relations under a probabilistic framework. Statistical models are often used to represent observations in terms of random variables. These models can then be used for estimation, description, and prediction based on basic probability theory. To increase the usefulness of qualitative spatial data it needs to be quantified, i.e., translating expressions such as "near" to actual distances. In our approach, we model a spatial relation between two POIs P_u, P_v in terms of *distance* and *orientation*. Our proposal is to use probabilistic modeling for this task, which includes the selection and extraction of respective features (distance and direction), as well as the methods to train and optimize the probabilistic model. Our work on mining and modeling geospatial knowledge from user generated textual data is presented in [SPK13] and [SPKS15].

1.2.2 Location Estimation

As we mentioned before, user-contributed content has benefited many scientific disciplines by providing a wealth of new data sources. However, the broad mass of users are much more comfortable generating *qualitative information*: People typically do not use coordinates to describe their spatial experiences (trips, etc.), but rely on qualitative concepts in the form of toponyms (landmarks) and spatial relationships (near, next, north of etc.). Thus, exploiting qualitative geospatial data is much more challenging as a geospatial data source.

Towards the second contribution of this dissertation, we consider supervised learning methods for quantifying qualitative data in order to solve the location estimation problem. Specifically, we want to estimate the position of unknown POIs in space based on their spatial relations with other known POIs. This is a very

important research area for two reasons: Firstly, the inherited noise in user generated content makes every geospatial problem and especially the location estimation problem quite challenging and important. Though, handling uncertainty in the geospatial domain should be thoroughly investigated. The second and most important motive is the fact that there are a lot of POIs with unknown coordinates. Most well known gazetteers contain about 10 million known POIs and "POIs generation", i.e, new POIs, on the web is a dynamic phenomenon. Consequently, we should provide approaches which can handle uncertainty and provide accurate estimates of POIs positions in space. Our work on the location estimation problem is presented in [SPKS15].

1.2.3 Popular Path Computation

Directions and paths, as commonly provided by navigation systems, are usually derived considering absolute metrics, e.g., finding the shortest or the fastest path within an underlying road network. With the aid of Volunteered Geographic Information (VGI), i.e., geo-spatial information contained in user generated content, we aim at obtaining paths that do not only minimize distance but also lead through more popular areas. Based on the importance of landmarks in Geographic Information Science and in human cognition, we employ the extracted and probabilistically modeled spatial relations from our previous contributions in order to solve the popular path computation problem.

The major challenge in this contribution is, (*i*) the extraction of crowdsourced geo-spatial information from textual data as explained before and, (*ii*) the enrichment of an existing road network with this information. The enriched road network is subsequently used to provide paths between a given start and target that satisfy the claim of higher popularity. Our work on the location estimation problem is presented in [SSJ+14], [JFS+15], [SSJ+15], [SJZ+15].

1.2.4 Mining GPS Data

Except textual data, GPS data are considered as a very rich source of UGC. Nowadays, massive amounts of tracking data for various types of moving objects, including vehicles, humans and animals, are becoming available. Analyzing this type of spatio-temporal data is crucial for discovering movement patterns, understanding and forecasting behaviors, and developing novel applications and services. One problem of particular interest is finding objects that move close together with a certain object during some periods of time.

In this dissertation, we focus on finding the k-nearest moving neighbors for a given query object and time interval. We attack the k-nearest moving neighbors for both the non-continuous and continuous query cases. For the non-continuous case, we formulate the problem, using a similarity function that takes into consideration both the proximity and the direction of the trajectories, and we propose some efficient exact and approximate algorithms. Our contribution is presented in [SSV13]. For the continuous case, we provide some baseline algorithms which apply for any aggregate function used to compute trajectory distances between objects and we continue and derive some optimized algorithms for the cases where the distance between two moving objects in a time window is determined by their maximum or minimum

distance in all contained timestamps and for the case that an upper bound on the velocities of the objects exists. Our on the continuous k-nearest moving neighbors problem is presented in [SSS14].

1.3 Outline

The outline of this work is as follows. Chapter 2 describes our contribution on mining spatial relations from user generated textual data and our probabilistic approach for modeling spatial relations. Chapter 3 describes our approach on the location estimation problem with extended real world and synthetic data experiments. Chapter 4 describes our contributions on the popular path computation problem. Moreover, we provide details about to demo implementations for the computation of popular paths on road networks. Chapter 5 describes our contributions on the non-continuous and continuous k-nearest moving neighbors problem. Finally, Chapter 6 of this dissertation provides conclusions and directions for future work.

Chapter 2

Mining and Modeling Geospatial Data

2.1 Mining Geospatial Information from UGC

2.1.1 Preliminaries

This chapter describes the processing tools and steps of our approach for spatial relation extraction from user generated texts.

In this work, we choose travel blogs as a potentially rich geospatial data source. This selection is based on the fact that people tend to describe their experiences in relation to their location, which results in "spatial narratives". To gather such data, we use classical Web crawling techniques [DP10] and compile a database[1] consisting of 250,000 texts, obtained from 20 travel blogs.

Obtaining qualitative spatial relations from text involves the detection of (*i*) spatial objects, i.e., Points-of-Interest (POIs) or toponyms, and (*ii*) spatial relationships linking the POIs. Our approach involves geoparsing, i.e., the detection of candidate phrases, and geocoding, i.e., linking parts of phrase/toponym to actual coordinate information.

Information Extraction (IE) is an important task in natural language processing, with many practical applications. It involves the analysis of text documents, with the aim of identifying particular types of entities and relations among them. Reliably extracting relations between entities in natural-language documents is still a difficult, unsolved problem especially when we are looking for spatial relations specifically. Its inherent difficulty is compounded by the emergence of new application domains, with new types of narrative that challenge systems developed for other, well-studied domains.

Traditionally, IE systems have been trained to recognize names of people, organizations, locations and semantic relations between them. The first and maybe most important part of this thesis, is the design and implementation of a system which extracts a specific type or relations between entities in textual data, namely spatial relations.

[1] Available upon request.

2.1.2 Related Work

The extraction of qualitative spatial data from texts requires the utilization of efficient NLP tools to automatically extract and map phrases to spatial relations. In the past, extraction of *semantic* relations between entities in texts is developed in [BM06], [FSE11], [AB09], [MSB13] and [ZAR02], while extraction of *spatial* relations (mostly topological relations) between entities in texts and web documents is analyzed in [KVOM11], [Yua11], [LIR+12], [ZZDZ11] and [WKB14].

In more detail, kernel methods, based mainly on a generalization of subsequence kernels for semantic relation extraction between entities in natural language text are developed in [ZAR02] and [BM06]. In [Yua11] a kernel based approach applied to Support Vector Machines (SVMs) is used to extract spatial relations from free text for spatial reasoning. SVM based extraction of spatial relations in text is addressed in [ZZDZ11], where the authors investigate the extraction of spatial relations based only on SVM models. The authors in [KVOM11] report on a novel task of spatial role labeling in natural language text. They propose machine learning methods to extract spatial roles and their relations. More specifically, they use a probabilistic approach by training Conditional Random Fields (CRFs), which is a special case of Markov Random Fields, in order to achieve spatial information extraction. Finally, extracting semantic relations from natural language text using dependency grammar patterns are addressed in [AB09]. The authors present Wanderlust, an algorithm that automatically extracts semantic relations from natural language text by using deep linguistic patterns.

While the above works constitute a good match in our developments for spatial relationship extraction from texts, we intentionally designed our specialized qualitative spatial data mechanism that better fits into the particularity, e.g., noisy crowdsourced data, of the relation extraction part of our problem. Our spatial relation extraction approach is based on the Natural Language Processing Toolkit (NLTK) which is presented in [LB02].

2.1.3 Contribution

Using the NLTK, which is a leading platform to analyze raw natural language data, we managed to extract 500,000 POIs from the text corpus. For the geocoding of the POIs, we rely on the GeoNames geographical gazetteer data, which covers all countries and contains over ten million place names and their coordinates. This procedure associates (whenever possible) geographic coordinates with POIs found in the travel blogs, using string matching based on the Levenshtein string distance metric [Hir97]. Using the GeoNames gazetteer, we managed to geocode about 480,000 out of the 500,000 extracted POIs.

The next step is the extraction of qualitative spatial relationships from text, which is a hard NLP problem. REVERB [FSE11] and EXEMPLAR [MSB13] are the state of the art available software tools for the extraction of semantic relations between identified entities in texts. However, they are generic NLP tools, not designed for spatial relations specifically. Moreover, we empirically observed that they perform poorly in several cases, when applied with a noisy crowdsourced dataset.

To this end, we design a more specialized spatial relation extraction algorithm, based on NLTK [LB02] components and predefined strings and syntactical patterns. Specifically, we define a set of language expressions, typically used to express a

spatial relation in combination with a set of syntactical rules. It turns out that the use of both syntactical and string matching reduces the number of false positives considerably. As an example, consider the following phrase. *"**Deutsche Bank** invested 10 million dollars **in Brazil**."*. Here, a simple string matching solution would extract a triplet of the form (Deutche Bank, in, Brazil), which is a false positive. In our approach, the use of predefined syntactical patterns avoids this kind of mistakes. On the other hand, for the phrase *"**Deutsche Bank** invested 10 million dollars **in Rio de Janeiro**, which is "**in Brazil**."* our algorithm would extract a triplet of the form (Rio de Janeiro, in, Brazil) which is a true positive.

Before we describe the architecture of our relation extraction approach, Table 2.1 shows indicative empirical results on a small dataset of 300 annotated crowdsourced spatial relations. Both our method and EXEMPLAR perform better than REVERB in terms of precision and recall. While our NLTK approach seems to have a slightly lower precision than EXEMPLAR [MSB13], it achieves a higher recall, where the fraction of extracted relations relevant to the query is larger.

Table 2.1: *Precision and recall for three different spatial relation extraction approaches.*

Method	Precision	Recall
EXEMPLAR [MSB13]	**0.7**	0.4
REVERB [FSE11]	0.1	0.4
NLTK [LB02]	0.6	**0.82**

2.1.3.1 Proposed scheme for relation extraction

The RELEX (Relation Extraction) algorithm (Algorithm 1), describes the architecture of the proposed information extraction system. Initially, the raw text document is segmented into sentences (Step 3). Each sentence is further subdivided (tokenized) into words and tagged as part-of-speech (Steps 5-6). Name entities (POIs) are identified (Step 7). We typically look for relations between specified types of named entities, which in NLTK are *Organizations, Locations, Facilities and Geo-Political Entities (GPEs)*. In sequence, in case there are two or more name entities in the sentence, we check if any of the predefined syntactical patterns applies between the recognized name entity pairs (Step 12). If so, we use regular expressions to determine the specific instance of the observed spatial relation from our predefined spatial relation pattern list (Step 14). If there is a string pattern match, we record the extracted triplet (Steps 15-18). Thus, the search for spatial relations in texts results into a set of triplets O of the form (P_u, R_o, P_v), where P_u and P_v are named entities of the required types and R_o is the observed spatial relation that intervenes between P_u and P_v.

A relation extraction example is shown in Figure 2.1. Here, sentence *"Boston is near New York"* is analyzed as explained, and two named entities are identified as GPEs.

We first check the syntax and the fact that the pattern *"GPE - 3rd person verbal phrase (VBZ) - preposition/subordinating conjunction (IN) - GPE"* exists in our set of predefined spatial relation patterns. Performing string matching on the intermediate chunks ("near") results in the triplet *(New York, Near, Boston)*.

ALGORITHM 1: RELEX - Spatial Relation Extraction

Input: A database of texts T, a set of syntactical patterns A, a set of spatial relation string patterns R

Output: A set of triplets O of the form $(P_u,\ R_o,\ P_v)$ where $P_u \neq P_v$ and $R_o \in R$

```
1  begin
2      Load syntactical 𝒜 and string ℬ patterns
3      foreach text t ∈ T do
4          Extract sentences from t into set S
5          foreach sentence s ∈ S do
6              Token s using NLTK
7              PosTag s using NLTK
8              Identify name entities using NLTK
9              if two or more name entities in s then
10                 Extract POI pairs in P
11                 foreach p ∈ P do
12                     p_A ← Extract syntactical pattern of p
13                     if p_A ∈ A then
14                         p_R ← Extract string pattern of p
15                         if p_R ∈ R then
16                             P_u ← p(1)
17                             P_v ← p(2)
18                             R_o ← p_R
19                             O.PushTriplet(P_u, R_o, P_v)
20                         end
21                     end
22                 end
23             end
24         end
25     end
26     return O
27 end
```

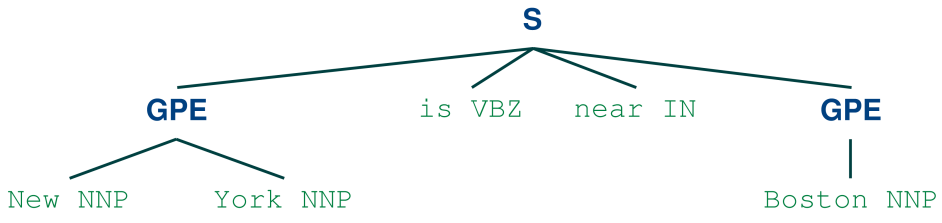
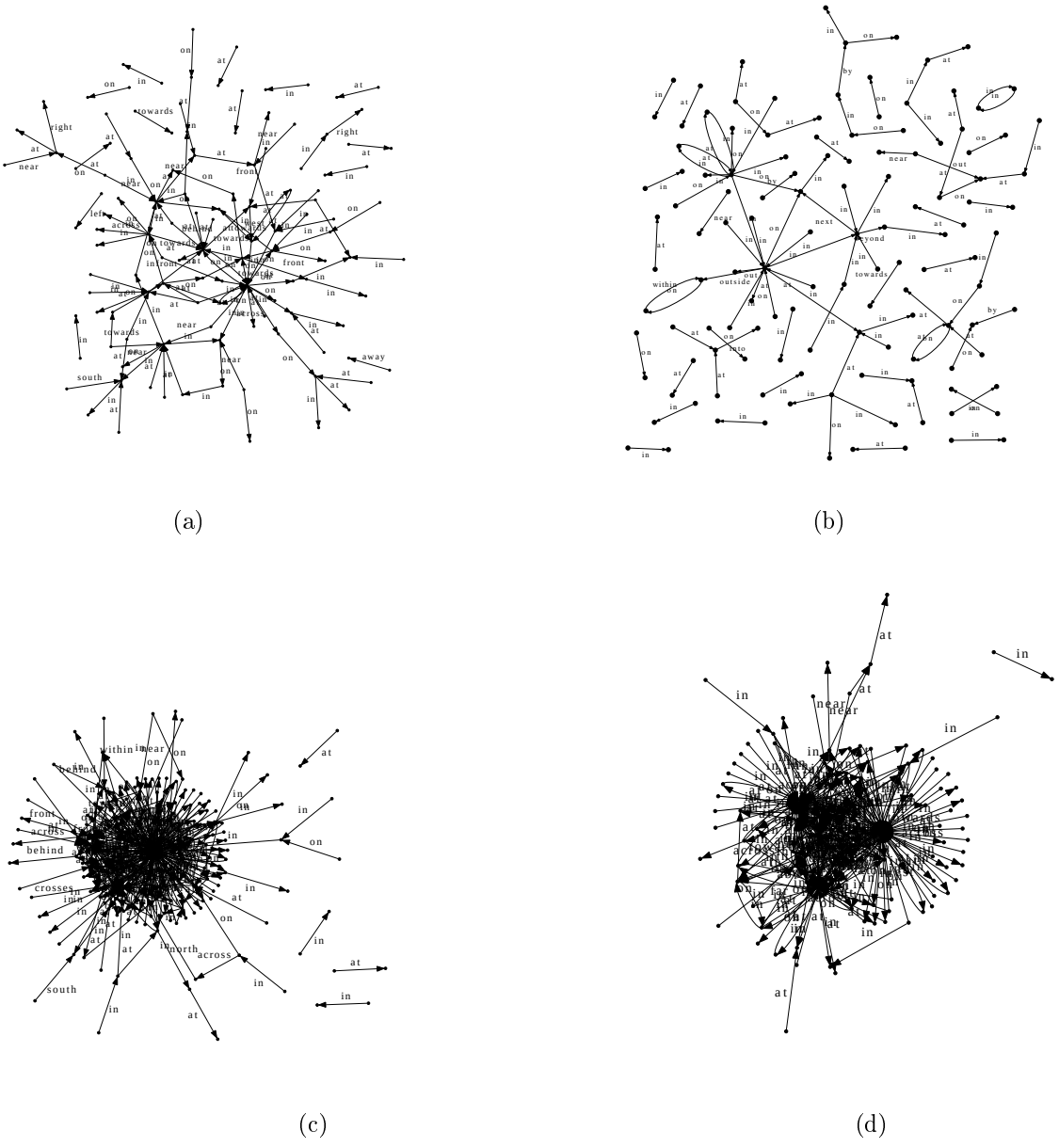


Figure 2.1: *Example of a parsed sentence syntactic tree.*

2.1.3.2 Spatial relation data

Applying Algorithm 1, we extracted 500,000 triplets from our 250,000 travel blog text corpus. Figure 2.2, shows small samples of *Spatial Relationship Graphs*, i.e., spatial graphs in which nodes represent POIs and edges label spatial relationships existing between them. The graphs visualize samples of the spatial relationship data collected for the cities of London, New York, Paris and Beijing respectively.

These four cases, going gradually from sparse to very dense spatial relation data, will be our main datasets during the experimental evaluation of the proposed approach.



(a) (b)

(c) (d)

Figure 2.2: *Small samples of spatial relationship graphs of (a) London (b) New York (c) Paris and (d) Beijing respectively.*

2.2 Modeling Spatial Relations

2.2.1 Preliminaries

Statistical models are often used to represent observations in terms of random variables. These models can then be used for estimation, description, and prediction based on basic probability theory. To increase the usefulness of qualitative spatial data it needs to be quantified, i.e., translating expressions such as "near" to actual distances. This chapter, presents a model for representing spatial relationships. In our approach, we model a spatial relation between two POIs P_u, P_v in terms of *distance* and *orientation*. Our proposal is to use probabilistic modeling for this task, which includes the selection and extraction of respective features (distance and direction), as well as the methods to train and optimize the probabilistic model.

Our analysis below includes, (*i*) feature extraction, (*ii*) an analysis of the probabilistic mixture models we employ for the quantitative representation of spatial relations and, (*iii*) a greedy learning algorithm for model parameter estimation. Overall, this section describes a method that trains probabilistic models for quantified spatial estimates of crowdsourced spatial relationships.

2.2.2 Related Work

The majority of works related to *qualitative approaches for spatial information representation* considers spatial relations. One popular spatial classification is constructed by topological relations (e.g., disjoint, overlap), direction relations (e.g., North, South), ordinal relations (e.g., inside, contain), and distance relations (e.g., far, near). The authors in [Ege89, EH90, ES93, KEG93] present formal methods for qualitative representation of spatial relationships based on mathematical theories of order. Their applicability on spatial database systems and some key-role technical concepts are coherently discussed in [GÖ94, PTS94, PSTE95]. Qualitative representation of spatial knowledge is discussed in [FGG+99, KSF+03, PS94]. The authors identify the common concepts of the qualitative representation and processing of spatial knowledge. They compare the representational properties of different systems and outline the computational tasks involved in relation-based spatial information processing. They also describe symbolic spatial indexes, relation-based structures that combine several ideas in spatial knowledge representation.

Recent research on *quantitative representation of spatial knowledge* has been conducted in relation to situational awareness systems, robotics, and image processing. Modeling uncertain spatial information for situational awareness systems is discussed in [KMM+06] and [MKM08]. The authors propose a Bayesian probabilistic approach to model and represent uncertain event locations described by human reporters in the form of free text. Estimation of uncertain spatial relationships in robotics is addressed in [SSC90]. A probabilistic algorithm for the estimation of distributions over geographic locations is proposed in [HE08] where a data-driven scene matching approach is used in order to estimate geographic information based on images. Finally, image similarity based on quantitative spatial relationship modeling is addressed in [WM03].

2.2.3 Contribution

2.2.3.1 Feature Extraction

In this contribution, we will model spatial relations based on distance and orientation features between POIs. Assuming a projected (Cartesian) coordinate system, the distance is computed as the Euclidean metric between the two respective coordinates, while the orientation is established as the counterclockwise rotation of the x-axis, centered at P_v, to point P_u.

For a concise and consistent mathematical formalization, consider that, for each instance of each relation, we create a two-dimensional *Spatial Feature Vector* $X = (X_d, X_o)^\intercal$ where X_d denotes the distance and X_o denotes the orientation between P_u and P_v. Several instances of a spatial relation lead to a set of two-dimensional spatial feature vectors which we denote as $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$. Each spatial feature vector set will be used to train one probabilistic model for each spatial relation.

An example of the feature extraction procedure is illustrated in Figure 2.3, where four instances of spatial relation *Near* are used in order to create the respective set of spatial feature vectors $\mathcal{X}_{near} = \{[X_{d1}, X_{o1}]^\intercal, [X_{d2}, X_{o2}]^\intercal, [X_{d3}, X_{o3}]^\intercal, [X_{d4}, X_{o4}]^\intercal\}$. In this scenario, $P_V = \{A, D, E, G\}$ is the set of reference points and $P_U = \{B, C, F, H\}$ is the set of points described based on the reference points.

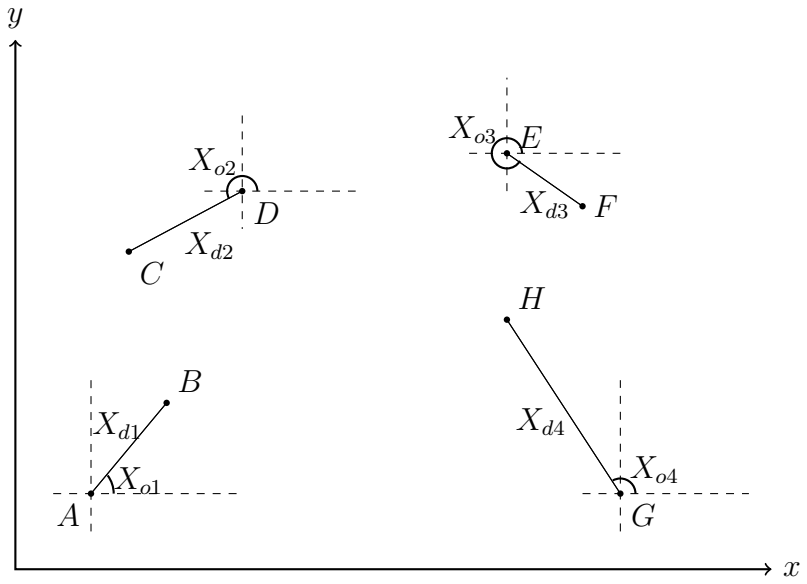


Figure 2.3: *Distance and orientation feature extraction procedure. In this case B is near A, C is near D, F is near E and H is near G.*

2.2.3.2 Probabilistic Modeling of Spatial Relations

The next step is the mapping of the data to pre-selected probability density functions (PDFs).

Given the training data, e.g., a set of spatial feature vectors $\mathcal{X}$ for each spatial relation, we fit a Gaussian Mixture model (GMM) for each one of these relations. The intuition is that people use spatial relation phrases in a different manner and to describe different POI positions. This results in multi-component distributions

of the features. Figure 2.4, illustrates a representative PDF example of distance and orientation features for spatial relation *"South"*, which strengthens further our intuition.

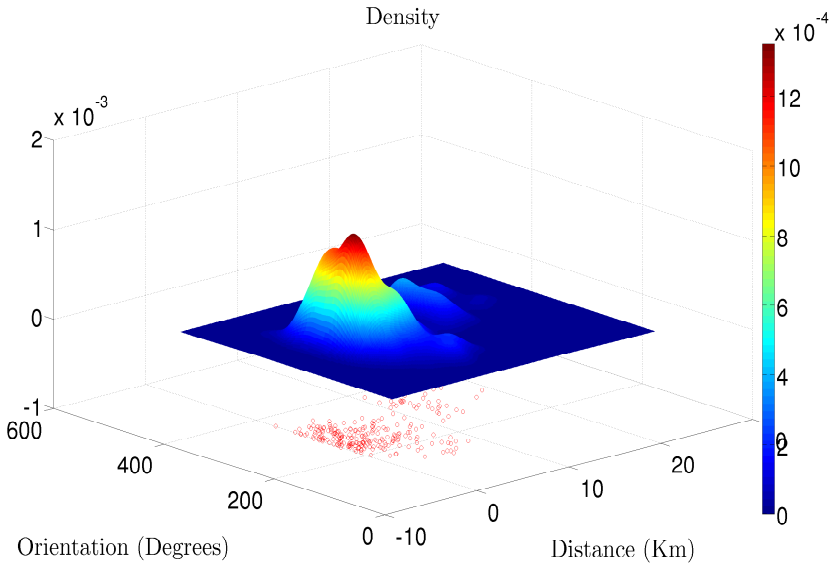


Figure 2.4: *Distance and orientation features for spatial relation South with the respective PDF.*

Moreover, in [LB99] it is shown that for any heterogeneous mutli-dimensional data that originates from an *arbitrary* PDF $p(\cdot)$, there exists a sequence of finite mixtures $p_k(x) = \sum_{i=1}^{k} w_i g(x;\theta_i)$ that achieves Kullback-Leibler (KL) divergence

$$D(p||p_k) - D(p||g_p) \leq \mathcal{O}(1/k)$$

for any $g_p = \int g(x;\theta)P(d\theta)$, i.e., one can achieve a good approximation with rate $\mathcal{O}(1/k)$ by using a k-component mixture of $g(x;\cdot)$. Furthermore, this bound is achievable by employing a *greedy training scheme* [LB99].

Finally, GMMs have been extensively used in many classification and general machine learning problems (cf. [Bis06] and [DHS01]). They are very well known for (i) their formality, as they build on the formal probability theory, (ii) their practicality, as they have been implemented several times in practice, (iii) their generality, as they are capable of handling many different types of uncertainty, and (iv) their effectiveness.

In general, a GMM is a weighted sum of M component Gaussian densities as $p(x|\lambda) = \sum_{i=1}^{M} w_i g(x;\mu_i,\Sigma_i)$, where x is a d-dimensional data vector (in our case $d = 2$), w_i are the mixture weights, and $g(x;\mu_i,\Sigma_i)$ is a Gaussian density function with mean vector $\mu_i \in \mathbb{R}^d$ and covariance matrix $\Sigma_i \in \mathbb{R}^{d\times d}$. To fully characterize f, one requires the mean vectors, the covariance matrices and the mixture weights. These parameters are collectively represented in $\lambda = \{w_i,\mu_i,\Sigma_i\}$ for $i = 1,\ldots,M$.

In our setting, *each spatial relation is modeled by a 2-dimensional GMM*, trained on each relation's spatial feature vector set. We assert that distance and orientation features are informative enough to model spatial relationships in a Cartesian context. For the parameter estimation of each Gaussian component of each GMM, we use *Expectation Maximization* (EM) (cf. [DLR77]). EM enables us to update the parameters of a given M-component mixture with respect to a feature vector

set $\mathcal{X} = \{X_1, \ldots, X_n\}$ with $1 \leq j \leq n$ and all $X_j \in \mathbb{R}^d$, such that the log-likelihood Equation 2.1 increases with each re-estimation step, i.e., EM re-estimates model parameters λ until $\mathcal{L}$ convergence.

$$\mathcal{L} = \sum_{j=1}^{n} \log(p(X_j|\lambda)) \tag{2.1}$$

The updates for the parameters of a GMM can be accomplished by iterative application of the following equations for all components $i \in \{1, ..., M\}$

$$P(i|X_j) = \frac{w_i g(X_j; \lambda_i)}{p(X_j|\lambda)} \tag{2.2}$$

$$w_i = \sum_{j=1}^{n} \frac{P(i|X_j)}{n} \tag{2.3}$$

$$\mu_i = \sum_{j=1}^{n} \frac{P(i|X_j)X_j}{nw_i} \tag{2.4}$$

$$\Sigma_i = \sum_{j=1}^{n} \frac{P(i|X_j)(X_j - \mu_i)(X_j - \mu_i)^\intercal}{nw_i} \tag{2.5}$$

The EM algorithm is not guaranteed to lead us to the solution yielding maximum log-likelihood on $\mathcal{X}$ among all maxima of the log-likelihood. Nevertheless, using the EM algorithm, if we are "close" to the global optimum (maximum) of the parameter space, then it is very likely we can obtain the globally optimal solution.

At this point, we highlight that the proposed scheme is distribution independent. With the necessary tweaks, one can transform the schema to use mixtures of *any distribution type* and such a selection is user-defined. However, its application would be much harder in practice. We use mixtures of pdfs, since it was shown [LB99] that densities of heterogeneous and noisy data, such as our case of crowdsourced data, can be approximated by a sequence of finite mixtures. We particularly use GMMs due to their simplicity and their generally low classification errors. Although proven to be not always the optimal choice, the results we obtain, and which we will thoroughly analyze in Section 5.2.4, encourage their use in practice. Thus, GMMs provide a challenging baseline for potentially better mixture models to be explored in future work.

2.2.3.3 Model Optimization

A main issue in probabilistic modeling with mixtures is that a predefined *number of components* is neither a dynamic nor an efficient and robust approach. The optimal number of components should thus be decided based on each dataset. Here, we employ a greedy learning approach to dynamically estimate the number of components in a GMM, as presented in [VVK03]. This approach, builds the mixture component in an efficient way by starting from an one-component GMM, whose parameters are trivially computed by using EM, and then employing the following two basic steps until a stopping criterion is met:

1. Insert a new component in the mixture

2. Apply EM until the log-likelihood $\mathcal{L}$ or the parameters of the GMM converge (cf. Section 2.2.3.2)

The stopping criterion can either be a maximum pre-selected number of components, or it can be any other model selection criterion. In our case the algorithm stops if the maximum number of components is reached, or if the new model's log-likelihood $\mathcal{L}+1$ is less or equal to the log-likelihood $\mathcal{L}$ of the previous model, after introducing a new component.

The greedy learning procedure can be summarized in Algorithm 2. For each spatial feature vector, we estimate the parameters and the log-likelihood of an one-component model (Steps 4-5). In sequence, we find a new component and add it to the previous mixture (Steps 7-8). Then, we re-estimate the model parameters and log-likelihood (Steps 9-10) until we reach the desiderata described above. The crucial step of this algorithm is the search for a new component (Step 7). Several approaches exist for this issue: One is to consider a number of candidates equal to the number of feature vectors but it is identified that such strategy would be rather expensive. The approach followed in this work is to pick an optimal number of candidate components. More specifically, for each insertion problem in a k-component mixture, the dataset $\hat{\mathcal{X}}$ is partitioned in k disjoint subsets and a fixed number m of candidate components is generated per existing mixture component, e.g., for a k-component mixture $k \times m$ candidate components are generated. In our experiments we used $m = 10$. Finally, using the EM algorithm, we pick the candidate component that maximizes the log-likelihood $\mathcal{L}+1$ when mixed into the previous mixture $p^M(\mathcal{X}|\lambda)$. The running time in order to learn a k-component GMM given n points is $O(nk^2)$.

The optimized GMM approach allows us to generate probabilistic models that quantify spatial relationships based on their observed use in textual narratives.

2.2.4 Similarity Between Quantitative Spatial Relationships

In many probabilistic classification problems several metrics have been proposed to compute a distance measurement between different classes as a means to compare them. Measuring distance between converged PDFs which model different classes (spatial relationships in our case) is a measure of similarity between them. In our contribution, we use Kullback-Leibler (KL) divergence [KL51] as such a distance metric.

There are two main reasons for checking similarity between quantified spatial relations. Firstly, we want to observe the changes for each GMM as we increase the maximum number of Gaussian components during the training procedure. Secondly, we use KL divergence to measure the similarity between spatial relationships that tend to follow similar patterns, e.g., *Near* & *NextTo*, *In* & *On*.

KL divergence is a similarity measure between two probability distributions. So, let $\mathcal{F}_1(x)$ and $\mathcal{F}_2(x)$ be two probability distributions (GMMs in our case). By definition, the KL distance $\mathcal{D}(\mathcal{F}_1(x)||\mathcal{F}_2(x))$ between $\mathcal{F}_1(x)$ and $\mathcal{F}_2(x)$ is given as follows.

$$\mathcal{D}(\mathcal{F}_1(x)||\mathcal{F}_2(x)) = \int \mathcal{F}_1(x) \log \left\{ \frac{\mathcal{F}_1(x)}{\mathcal{F}_2(x)} \right\} dx \tag{2.6}$$

ALGORITHM 2: Optimized GMM Training

Input: A set of spatial feature vectors $\hat{\mathcal{X}}$, a maximum number of components in $\mathcal{M}_{\mathcal{C}}$

Output: A set of trained GMMs $\hat{\mathcal{G}}$

begin
  $M \leftarrow 1$
  foreach $\mathcal{X} \in \hat{\mathcal{X}}$ **do**
    $p^M(\mathcal{X}|\lambda) \leftarrow$ Estimate 1-component model parameters using EM
    $\mathcal{L}^M \leftarrow$ Calculate 1-component model log-likelihood
    while $M \leq \mathcal{M}_{\mathcal{C}}$ **do**
      $g(\mathcal{X}; \lambda^*) \leftarrow$ Optimal new component for $(p^M(\mathcal{X}|\lambda))$
      $p^{M+1}(\mathcal{X}|\lambda) \leftarrow$ Combine model $p^M(\mathcal{X}|\lambda)$ and component $g(\mathcal{X}; \lambda^*)$ in a new model
      $p^{M+1}(\mathcal{X}|\lambda) \leftarrow$ Estimate new model parameters using EM
      $\mathcal{L}^{M+1} \leftarrow$ Calculate new model log-likelihood
      if $\mathcal{L}^{M+1} \leq \mathcal{L}^M$ **then**
        $\hat{\mathcal{G}}.PushGMM(p^M(\mathcal{X}|\lambda))$
        $Terminate()$
      else
        $M \leftarrow M + 1$
      end
    end
    $\hat{\mathcal{G}}.PushGMM(p^M(\mathcal{X}|\lambda))$
  end
  return $\hat{\mathcal{G}}$
end

The KL divergence is always nonnegative and it is zero only when the two distributions are identical. Additionally KL divergence is not symmetric, i.e., $\mathcal{D}(\mathcal{F}_1(x)||\mathcal{F}_2(x)) \neq \mathcal{D}(\mathcal{F}_2(x)||\mathcal{F}_1(x))$. It is common to encounter the symmetric version of the KL divergence between $\mathcal{F}_1(x)$ and $\mathcal{F}_2(x)$) as

$$\mathcal{D}_{sym}(\mathcal{F}_1(x)||\mathcal{F}_2(x)) = \frac{\mathcal{D}(\mathcal{F}_1(x)||\mathcal{F}_2(x)) + \mathcal{D}(\mathcal{F}_2(x)||\mathcal{F}_1(x))}{2} \tag{2.7}$$

In this work, we use the symmetric KL divergence in order to measure the similarity between GMMs.

2.2.5 Experimentation

The scope of this section is to assess the quantitative representation of qualitative geospatial data by means of probability distributions (GMMs). For this purpose, we investigate a set of spatial relationships for a specific geographic area (London). In terms of experiments, we compute probabilistic representations of spatial relationships by considering distance and orientation as dependent but, uncorrelated features (case one) and as correlated features (case two).

We visualize the results of the trained models and compare them to check if they intuitively perform well, e.g., they return visually reasonable results. In addition, we measure the KL divergence for spatial relationships between a baseline one-component model and the maximum number of Gaussian components model. Finally, based on visualization and KL divergence, we assess the informativeness and efficiency of distance and orientation features for quantitative modeling of spatial relations and observe how much different spatial relations may behave in a similar way.

2.2.5.1 Experimental Setup

The choice of an appropriate dataset is crucial in our experimentation. As mentioned is Section 2.1.3.2, the density of POIs is very high in urban regions. We decided to use data from such a dense region to find meaningful as well as consistent spatial relationships. We retrieved data for a bounding box that contains the greater area of London, UK. In this preprocessing step, we parsed our travel blog data (120k texts) set and retrieved sentences containing at least two POIs and whose coordinates are within the bounding box of Latitude $[51°, 52°]$ and longitude $[-1°, 1°]$. This resulted in 12k sentences. Using human annotation, we extracted instances of the eight most frequent spatial relations including *North, South, East, West, Near, In, On, NextTo*. This means also that in our travel blog dataset, people tend to use a mixture of directional, topology and vague metrical relations in order to describe POI locations. From this data, distance and orientation features where extracted as described in Section 2.1.3.2.

Next, we employ the greedy EM algorithm to train bivariate GMMs based on the extracted distance and orientation features for each spatial relationship. The results are PDFs for each spatial relationship that, as the initial outset suggests, can be used to estimate the unknown position of spatial objects.

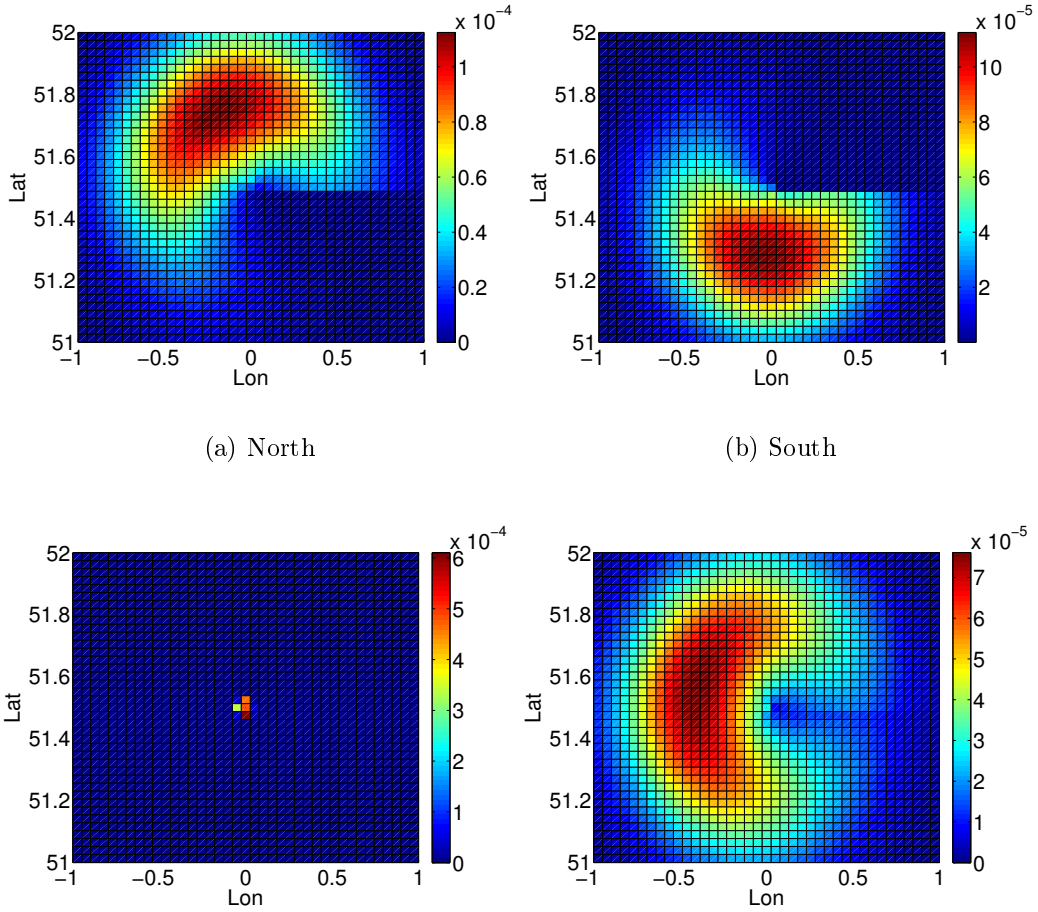
Our approach has been implemented in Matlab and all the experiments were conducted on an Intel(R) Core(TM) i5-2400 CPU at 3.10GHz with 8GB of RAM, running Ubuntu Linux 11.10.

2.2.5.2 Visualization of Quantitative Spatial Relations

The most important means of assessing the result is to visualize the quantified spatial relations. We divided the London bounding box to filter the input data by means of a 50×50 spatial grid. Each grid cell corresponds to a $4.4km \times 2.2km$ spatial extent (Longitude, Latitude). Given two spatial objects and the known location at the center of the grid, we plot for each grid cell the positional probability of the unknown location, i.e., how likely it would be for the unknown spatial object to be located in a specific grid cell. Using a heat map, warmer colors (red) indicate higher probabilities.

Figure 2.5 shows four spatial relationships modeled as one-component GMMs, with distance and orientation considered as uncorrelated random variables.

The proposed modeling based on distance and orientation features performs especially well in some of the cases. More specifically, for the cases of *North* (cf. Figure 2.2.5.2), *South* (cf. Figure 2.2.5.2) and *Near* (cf. Figure 2.2.5.2) the proposed model returns high probabilities in the expected regions. On the other hand, the case of *In* (cf. Figure 2.2.5.2) seems to include a lot of statistical noise due to



(a) North

(b) South

(c) Near

(d) In

Figure 2.5: *Probabilistic heat maps for four basic spatial relationships: 1-Component Gaussian Mixture Models for the uncorrelated distance and orientation case. All figures illustrate the case where a POI is conntected with the center of the grid, with the respective spatial relation.*

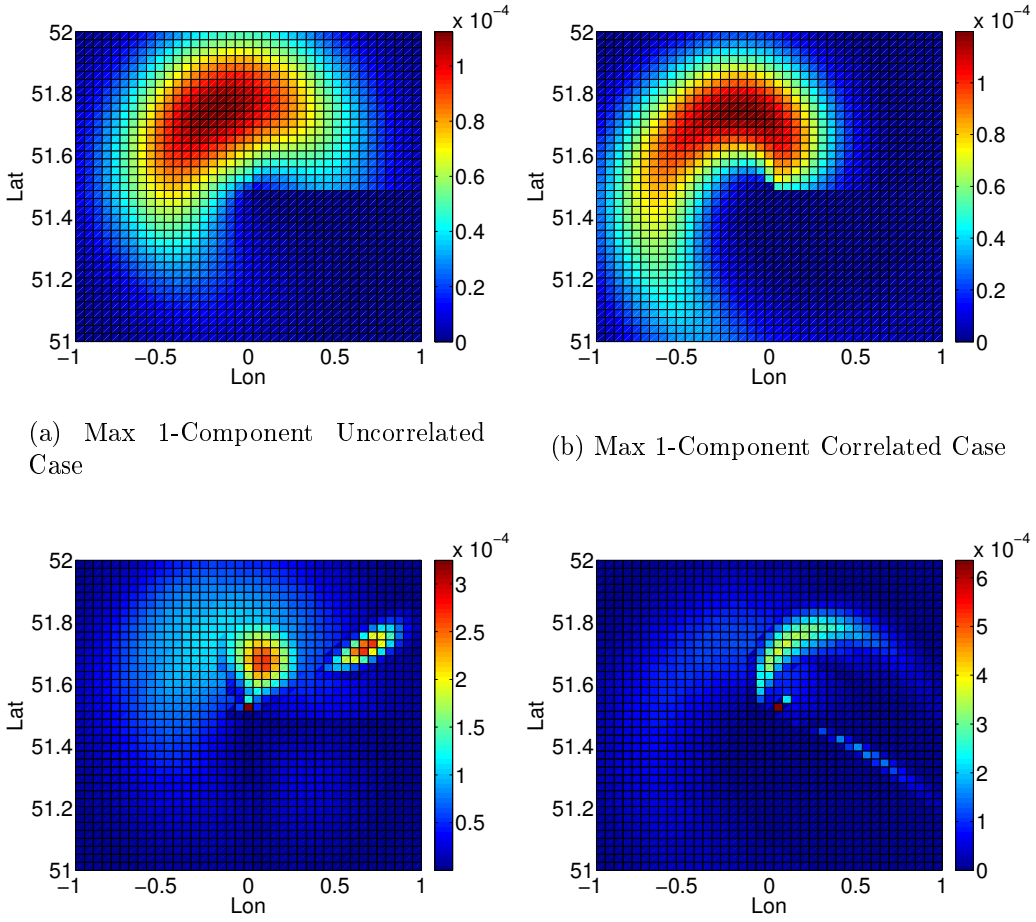
the general uncertain nature of user generated content. For example, high distance and orientation variance values for the cases of *On* and *In* are caused by the the fact that most of the sentences that contain these spatial relations are of the form ***POI*** *in London* and ***POI*** *on river Thames*.

2.2.5.3 Optimal Number of Gaussian Components

An important parameter when generating GMMs is the maximum number of Gaussian components. Such a limit is simply a stop criterion in the GMM training process and does not mean that the final component will converge to this upper limit, e.g., it might already converge to a lower number of components. Figure 2.6 illustrates the case of spatial relation *North*. The heat maps of Figures 2.2.5.3 and 2.2.5.3 show the cases of a maximum of 1 Gaussian components per mixture when distance and orientation are considered as uncorrelated and correlated, respectively. The heat maps of Figures 2.2.5.3 and 2.2.5.3 show the cases of a maximum of 5 Gaussian components per mixture, when distance and orientation are considered as

uncorrelated and correlated, respectively.

For both uncorrelated and correlated cases, Figures 2.2.5.3 and 2.2.5.3 show that by stepwise increasing the maximum number of Gaussian components, high probabilities tend to accumulate in fragmented small regions. The reason is that higher number of mixtures per GMM leads to components that are converging on their parameters (mean, covariance, component weight) based on more dense regions of the dataset, e.g., regions with more data samples will become dominant components. The weight of such *dominant components* (it is denoted as w in previous Section) is higher than the weight of other components in the final GMM. This results in fragmented high probability regions in the final heat maps.



(a) Max 1-Component Uncorrelated Case

(b) Max 1-Component Correlated Case

(c) Max 5-Component Uncorrelated Case

(d) Max 5-Component Correlated Case

Figure 2.6: *Probabilistic heat maps for* North*: (a), (b) show correlated and uncorrelated distance and orientation case for a max of 1 Gaussian component per GMM while (c), (d) show correlated and uncorrelated distance and orientation case for a max of 5 Gaussian components per GMM.*

As a result of this phenomenon, a major question is to the best approach to decide about the number of components per GMM. From an intuitive point of view, a smaller number of Gaussian components performs better as it preserves spatial generality, i.e., trends. However, as high probability regions are larger, they might result in inefficient location prediction tasks. From a statistical point of view, a

high number of mixture components results in more accurate probabilistic models and better classification performance in most of the cases. Unfortunately, the latter approach leads to sparse and small, high probability regions, which could be characterized as biased to the specific characteristics of the geographic region from where the dataset is taken (London in our case).

In Figures 2.2.5.3 and 2.2.5.3, we depict the *average log-likelihood*, e.g., we estimated parameters and log-likelihoods for each spatial relation model running the greedy learning algorithm 100 times per maximum component step and stepwise increasing the maximum Gaussian components per GMM. Figures 2.2.5.3 and 2.2.5.3 show the cases of correlated and uncorrelated distance and orientation random variables, respectively. In both cases, most of the spatial relation models converge on a high number of components, i.e., 16-17. Only spatial relations *Near* and *NextTo* converge on a smaller number of components. This means that statistically, most of the spatial relationships should be modeled with an upper limit of Gaussian components close to 16 or 17. In practice, this will result in fragmented spatial probabilities (heat maps) as outlined above.

Concluding, we realize that based on the log-likelihood measurements, there are statistically correct and sometimes optimal solutions for deciding the number of components. The difficult part in our case is the balance between statistical and intuitive robustness.

Based on a user generated dataset, we believe that GMMs with a number of components between 1 and 10 are statistically correct (but not optimal) and intuitively efficient to model spatial relations.

2.2.5.4 Correlated vs. Uncorrelated Feature Vectors

Correlation between distance and orientation is another important issue when training GMMs. Literature suggests that most of the classification approaches perform better when probabilistic models are trained taking into consideration the correlation between random variables. In our work, visualization shows that there is a high correlation between distance and orientation for some but not all cases. Figure 2.6 illustrates the case of *North*. For the heat maps shown in Figures 2.2.5.3 and 2.2.5.3, distance and orientation are considered uncorrelated, for Figures 2.2.5.3 and 2.2.5.3, they are considered as correlated random variables. Intuition suggests that we can not guarantee that the correlated case performs better, even if we are sure that distance and orientation are correlated. The *North* case should result in high probabilities for the top part of the grid as it should be the case for all directional relations. However, based on visual results and heat maps for all modeled spatial relations, we observe that distance and orientation seem to be less correlated for the cases of *In* (cf. Figure 2.2.5.2) and *On*, and tend to have zero correlation, e.g., region around the center of the grid with almost equal probabilities, for the cases of *Near* (cf. Figure 2.2.5.2) and *NextTo*. As expected, this leads as to the conclusion that some spatial relations are independent of orientation, e.g., only distance could model them efficiently. This also means that distance and orientation should be modeled as independent random variables.

Summing up, based on user-generated content, we believe that directional relations like *North* should be modeled taking correlation between distance and orientation into consideration. On the other hand, topological relations such as *In* and metric relations like *Near* tend to be independent of orientation, which means that

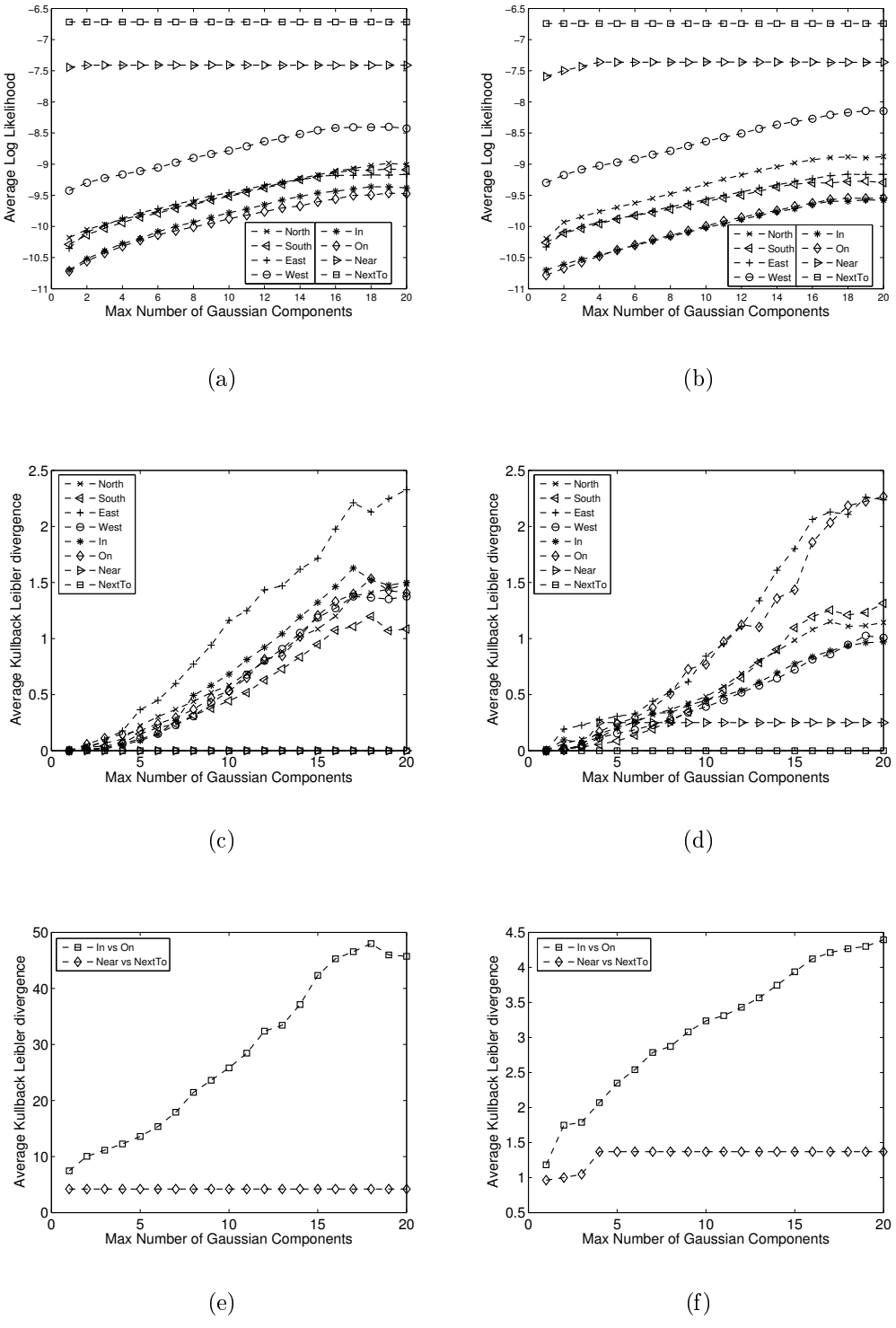


Figure 2.7: *(a), (b) Average log-likelihood vs maximum number of Gaussian components for correlated and uncorrelated distance and orientation case respectively. (c), (d) Average KL divergence between the baseline 1-component GMM and the final converged GMM after each step of increasing the maximum number of components for correlated and uncorrelated distance and orientation case respectively. (e), (f) Average KL diverge between spatial relationship pairs "In-On" and "Near-Nextto" for correlated and uncorrelated distance and orientation case respectively.*

correlation between distance and orientation should not be taken into consideration during modeling.

2.2.6 Similarity Between Quantified Spatial Relations

Besides a visual inspection, it is important to have a quality metric to assess the probabilistic spatial relation quantification. We use Kullback-Leibler (KL) divergence (i) to assess the similarity between converged GMMs of the same relation and (ii) to measure the similarity between some spatial relationships that tend to follow similar patterns.

Figures 2.2.5.3 and 2.2.5.3 illustrate the KL divergence between the baseline 1-component GMM and the final converged GMM after each step of increasing the maximum number of components for correlated and uncorrelated cases, respectively. Most of the models tend to diverge from the baseline model as we increase the maximum number of components. Only the models for *Near* and *NextTo* have low and zero distance from their baseline model. This matches the corresponding log-likelihoods illustrated in Figures 2.2.5.3 and 2.2.5.3, which remain almost stable. In these examples, with a small number of Gaussian components for *Near* and one Gaussian component for *NextTo* the log-likelihoods remains stable.

Finally, Figures 2.2.5.3 and 2.2.5.3 show that spatial relation pairs *Near-NextTo* and *On-In* exhibit similar characteristics. To assess this similarity, we measured the KL divergence for all cases of their models. The aforementioned figures show that the pair *On-In* seems to diverge as we increase the number of components for both correlated and uncorrelated cases. However, the pair *Near-NextTo* exhibits low values of KL divergence for all cases. This leads to the conclusion that people use more than one language expression to describe the same spatial relation. For our examples, this means that we could merge the cases of *Near* and *NextTo* into one probabilistic model.

Chapter 3

Location Estimation

3.1 Preliminaries

Off-the-shelf geospatial information services are typically based on *quantitative*, coordinate-based data: maps are generated to answer geospatial questions such as "Where is the Monastiraki Metro Station (Athens)", based on their accurate descriptive information; i.e., if such information appears in the pre-compiled database, this implies we know—within some accuracy—the coordinates of the Monastiraki Metro Station. Such geospatial information services have been proved to be extremely useful tools across many disciplines, from natural research management to transportation planning and from public information services to forest types mapping.

One of the reasons for their success is the *accuracy of information provided:* built by a community of mappers that contribute and maintain geospatial data, they provide precise answers to geographic queries based on coordinate information. However, the generation, process and preservation of such quantitative data is cost- and time-ineffective: While technology has helped to facilitate such geospatial data collection (e.g., all smart phones are equipped with GPS positioning sensors), yet authoring quantitative data requires constant supervision and control in order to preserve a quality of service.

On the other hand, user-contributed content has benefited many scientific disciplines by providing a wealth of new data sources. However, the broad mass of users are much more comfortable generating *qualitative information*: People typically do not use coordinates to describe their spatial experiences (trips, etc.), but rely on qualitative concepts in the form of toponyms (landmarks) and spatial relationships (near, next, north of etc.). Thus, exploiting qualitative geospatial data is much more challenging as a geospatial data source.

In this paper, *we consider supervised learning methods for quantifying qualitative data* in order to solve the following problem:

PROBLEM: *Given a set of objects P_V with a-priori known coordinates in space, a set of objects P_U whose exact positions are unknown and a set of predefined spatial relationships R between P_U and P_V objects, find probabilistic estimates for the positions of P_U objects in space.*

To provide some intuition, consider the following narrative: *"The best pita place is **next to** the Monastiraki Metro Station in Athens."* One of the challenges here is

the uncertainty associated with this sentence: The same concept ("next to") might be interpreted differently by the various users. For example, it is apparent that the relation "next to" might not imply any orientation (i.e., "west to", "east to", etc.). Nevertheless, given a predefined grid of points over and around the Monastiraki Metro station, we desire to associate probabilities to each location on the grid as candidate positions of the best pita place. The probabilities assigned are drawn according to probabilistic models, trained and learned using narratives including "next to" relation in a region relatively close to the point of interest (POI). I.e., we want to quantify what people imply when they say *"next to"*. Being able to do so, might allow us to actually discover the "best pita place in Athens"; see Figure 3.1 for a toy example explanation. Eventually, by collecting more observations that mention the "best pita place in Athens" using qualitative spatial information, we will be able to refine the unknown location and, thus, locate places that otherwise could not be geocoded.

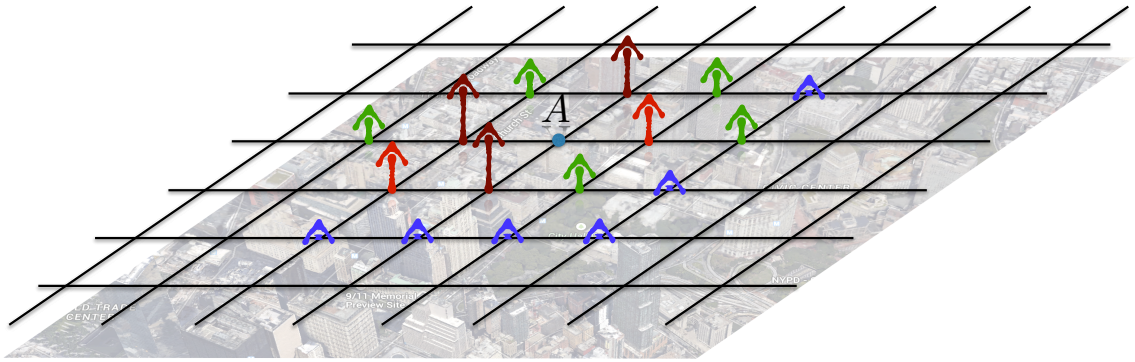


Figure 3.1: *Here, point A corresponds to known POI in downtown, New York, and the arrows on the predefined grid locations correspond to probabilities of spots being the "best restaurant near to" point A, as learned from training on a corpus of data. The higher the arrow, the higher the probability the location of interest lies at the corresponding grid point. Our aim is to locate the best restaurant from such probability measures on a fine grid of the space.*

From the above discussion, it is obvious that the problem at hand contains high uncertainty, especially when the source of spatial information is user-contributed. Our approach follows a probabilistic path, where we quantify qualitative relations as probability measures. Using textual narratives, we first learn spatial relation models between known POIs: i.e., focused over a predefined region of interest (i.e., Athens) and given POI pairs with known locations and linked together by a specific spatial relationship, we train the corresponding spatial relation pdf, as presented in 4.3.2, comprised of distance and orientation. To do so, a greedy Expectation Maximization-based (EM) algorithm is used. The trained probabilistic models can then be used for location estimation tasks.

Given a specific spatial relation instance of the form (P_u, R_o, P_v) and by employing the trained model for spatial relation R_o, we can associate positional probabilities for each point on the discretized space. These positional probabilities are then used to "triangulate" the positions of unknown POI locations. The more observations we have with respect to an unknown location, the more precise we will be in unknown POI's localization. Actual location estimation experiments using textual narratives

from travel blogs, establish the validity and quality of the proposed approach.

Our **contributions** can be summarized as follows:

(*i*) We employ a natural language processing (NLP) tool based approach that automatically extracts qualitative VGI (POIs and spatial relations) crawled from travel blogs as presented in Chapter 2.

(*ii*) We quantify qualitative spatial relations using a probabilistic path as presented in our previous work (c.f. [SPK13]) as presented in Chapter 2.

(*iii*) We propose a grid based algorithm which performs location estimation based on the aforementioned probabilistic models for spatial relations.

(*iv*) We evaluate our location prediction algorithm with extended experiments on both synthetic and real world location prediction scenarios.

3.2 Related Work

Work relevant to this part of the dissertation includes: (*i*) location estimation of multimedia data and Twitter users and (*ii*) location estimation of unknown points of interest from textual data.

3.2.1 Location Estimation of Multimedia Data and Twitter Users

[FVD10] is one of the first attempts for multimodal location estimation on videos where visual, acoustic and textual information is combined in order to declare where a video was recorded. Furthermore, [CLE+13] extends this work where the authors study human performance as baseline for location estimation for three different combinations of modalities (audio only, audio + video, audio + video + textual metadata) and compares it with the machine algorithm's performance in [FVD10]; the study demonstrates cases when humans could effectively identify audio cue for estimating video's location when the machine algorithm failed. [KSC+13] combines the data from the visual and textual modalities with external geographical knowledge bases by building a hierarchical model that combines data-driven and semantic methods to group visual and textual features together within geographical regions. As a result, the proposed method successfully located 40% of the videos in the MediaEval 2010 Placing Task test set within a radius of 100m.

From a different perspective, the authors in [CCL10] consider geo-location prediction from Twitter data. In particular, the authors propose a probability framework to estimate city-level location of a Twitter user based on tweet content. According to their results, about half of the Twitter users can be placed within 100 miles of their true locations. Following this line of work, the authors in [CLEL12] propose to model the spatial usage of a word as a Gaussian mixture model; an approach that is also followed in our work. Content-based machine learning techniques for Twitter user localization are presented in [JPS13], while the authors in [HCB12] and [HCB14] study the location estimation problem which is based on the automatic

identification of location indicative words: that is words that implicitly or explicitly encode an association with a particular location.

We highlight that, while the above approaches study a similar problem to ours, they do not handle scenarios where the observations contain POIs with positions not stored in a geographical database. In stark contrast, our approach can further improve upon the papers above by providing probabilistic location estimates for POIs observed for the first time.

3.2.2 Location Estimation of Unknown Points of Interest

While the present paper was written, we became aware of independent recent works on *location estimation of points of interest that are not stored in any geographical database*. Specifically, the authors in [MGM14] and [MRANIG14] propose an unsupervised geocoding algorithm which employs clustering techniques in order to estimate a spatial footprint of toponyms not found in gazetteers. The authors evaluate their approach with a corpus of real hiking descriptions in three different languages. Yet, there is an important difference with our setting: the authors assume that there is no uncertainty included in human descriptions, a rather strong assumption for real applications. In particular, they consider the hiking descriptions as a-priori 100% correct and they provide a heuristic solution based on predefined patterns and categories of spatial relationships. We believe that our approach, i.e., probabilistic modeling of spatial relationships, is considered as an important complementary feature in such scenarios that can effectively handle the uncertainty contained in user generated texts and further improve other proposed approaches for location prediction of unknown POIs.

3.3 Contribution

Spatial relationships are essentially observations between spatial objects. Having quantified them allows us to reason about locations. Our goal now is to show how such probabilistic models can be employed in location estimation scenarios. Unknown locations can be estimated by fusing spatial relationship observations to known POIs (landmarks). The QLEST (Qualitative Location Estimation) algorithm (Algorithm 3) details our location estimation method for a given set of unknown POIs P_U and a given set of triplets $\mathcal{T}$ of the form (P_u, R_o, P_v) (where P_v is a known landmark POI) about each POI in P_U. QLEST is a grid based approach, where given a predefined grid of points over and around a landmark POI P_v, we desire to associate probabilities to each vertex on the grid as candidate positions of the unknown POI P_u. The probabilities assigned are drawn according to probabilistic models, trained and learned as described in Section 4.3.2. Finally, all the assigned probabilities are aggregated in order to define the overall probability of each grid cell.

Specifically, the first step is to discretize space by partitioning it with respect to grid vertices (landmarks) (Step 2). For example, for a grid dimensionality $\mathcal{G}_\mathcal{D} = 15$, we have $15 \times 15 = 225$ grid vertices and $14 \times 14 = 196$ grid cells (regions). For each unknown POI P_u , we load the respective triplets $\mathcal{T}$ of the form (P_u, R_o, P_v) (Step 5).

ALGORITHM 3: QLEST - Qualitative Location Estimation

Input: A set of trained GMMs $\hat{\mathcal{G}}$, a bounding box $\mathcal{B}_{\mathcal{B}}$, grid dimensionality in $\mathcal{G}_{\mathcal{D}}$, a set P_U of POIs to locate, a set P_V of landmark POIs P_V, and a set $\mathcal{TT}$ of sets $\mathcal{T}$, i.e., $\mathcal{T} \subset \mathcal{TT}$, of triplets ($P_u$, R_o, P_v)

Output: Estimation accuracy $\mathcal{A}$

```
1  begin
2      G_V ← Calculate grid vertices for B_B based on G_D
3      InTopK ← 0
4      foreach P_u ∈ P_U do
5          T ← Load all triplets for PoI P_u from TT
6          Indx1 ← 0
7          foreach tu ∈ T do
8              Ĝ' ← Load GMM for spatial relation R_o of triplet tu
9              Indx2 ← 0
10             foreach gv ∈ G_V do
11                 V_L(Indx1,Indx2) ← Calculate each gv vertex's likelihood
                   with P_v of triplet tu as reference point, given model Ĝ'
12                 Indx2 ← Indx2 + 1
13             end
14             Indx1 ← Indx1 + 1
15         end
16         F_VL ← Sum and normalize each vertex's likelihoods in V_L
17         R_L ← Calculate each region's likelihood using F_VL
18         if P_u in K highest probability R_L then
19             InTopK ← InTopK + 1
20         end
21     end
22     A ← Percent(InTopK)
23     return A
24 end
```

Continuing, for each triplet tu we load the GMM relationship model $\hat{\mathcal{G}}'$ which corresponds to spatial relation R_o (Steps 7-8). Then, the selected relationship model $\hat{\mathcal{G}}'$ is used to assign positional probabilities to each vertex (landmark) gv of the grid, using P_v of triplet tu as a reference point (Steps 10-13). In this way, we update the likelihoods of all vertices in the grid. All likelihoods are stored in matrix $\mathcal{V}_{\mathcal{L}}$ (Step 11). Finally, all likelihoods per vertex are summed up and normalized in $\mathcal{F}_{\mathcal{VL}}$ (Step 16) and *a probability is assigned to each grid cell* (region) $\mathcal{R}_{\mathcal{L}}$ (Step 17). The overall likelihood of a grid cell is calculated as the mean value of the likelihoods of its four vertices. In a final step, we keep track of how many times the region that contains the unknown point is ranked among the k-highest (Top-k) probable regions (Steps 18-20). This allows us to measure the estimation accuracy of the proposed approach (Step 22).

The presented QLEST method, which fuses (spatial relationship) observations to estimate unknown point locations, will be used in the following experimental section

in the context of synthetic and real-world location estimation scenarios.

3.4 Experimentation

To assess the quality of the probabilistic spatial modeling and location estimation approach, we perform the following extensive experimentation using synthetic and real-world location estimation scenarios. All text processing has been implemented in Python, while the relationship modeling and location estimation methods where implemented in Matlab.

3.4.1 Location Estimation for Synthetic Scenarios

In order to provide some baseline results, we define a *1-component baseline* (BSL) model (which is a GMM model $p(x|\lambda) = \sum_{i=1}^{M} w_i g(x; \mu_i, \Sigma_i)$ with $M = 1$), and an optimized model (OPT), trained as described in Section 2.2.3.3 and using Algorithm 2.

For the synthetic location estimation scenarios, we have to generate POIs and spatial relations that connect them with known landmarks. Therefore, we follow a similar path as in Algorithm 3. Firstly, we discretize space by partitioning it with respect to grid vertices ($\mathcal{G}_\mathcal{V}$), which will be used as landmarks, and we generate a random point (P_u). Continuing, for each grid vertex ($gv \in \mathcal{G}_\mathcal{V}$) we pick the spatial relationship GMM ($\hat{\mathcal{G}}'$) that maximizes the likelihood—in terms of probability—of P_u. Under a mathematical formalization this means that $\hat{\mathcal{G}}'$ is picked as $\hat{\mathcal{G}}' \leftarrow \arg\max_{g \in \hat{\mathcal{G}}} P(P_u|g, gv)$, where $\hat{G}$ is again a set of trained spatial relation GMMs. Thus, for each random point P_u we generate equal number of triplets of the form (P_u, R_o, P_v) with the number of grid vertices with P_v being always a grid vertex.

Following this procedure, we generate 1000 location prediction scenarios for each of our four datasets. Figure 3.2 illustrates the approach by means of the (very challenging) example of Beijing. The light gray colored grid cell in Figure 3.4.1 illustrates the region in which a random point was generated. Figure 3.4.1 shows the assigned probabilities of each region after a full run of Algorithm 3. We observe that our approach assigns the highest probability to a location by using only spatial relationships extracted with respect to the landmarks in the region. Figure 3.4.1 illustrates the monitoring of the log-likelihood of the random point region as we sequentially visit each vertex (landmark) in the grid. Starting from the lower left grid vertex ($\hat{\mathcal{G}}'$ model 1) and proceeding row-wise until the upper right vertex ($\hat{\mathcal{G}}'$ model 225), the *log-likelihood increases as we move closer to the desired region* and decreases when moving away. Figure 3.4.1 points out the five highest likelihoods of the random point region and Figure 3.4.1 the locations of these vertices (landmarks) in the grid along with the model's textual description. This analysis approach should be considered also a qualitative accuracy assessment. Most of the models selected are qualitatively correct and express a real spatial relation between each corresponding vertex and the random point.

Additionally, we consider the cases in which the randomly generated point's region is among the Top-k predicted regions with $k = \{1, 5, 10, 20\}$, respectively. The prediction accuracy results are shown in Figure 3.3. The results show the superior performance of the OPT model. Additionally, Table 3.1 shows the actual

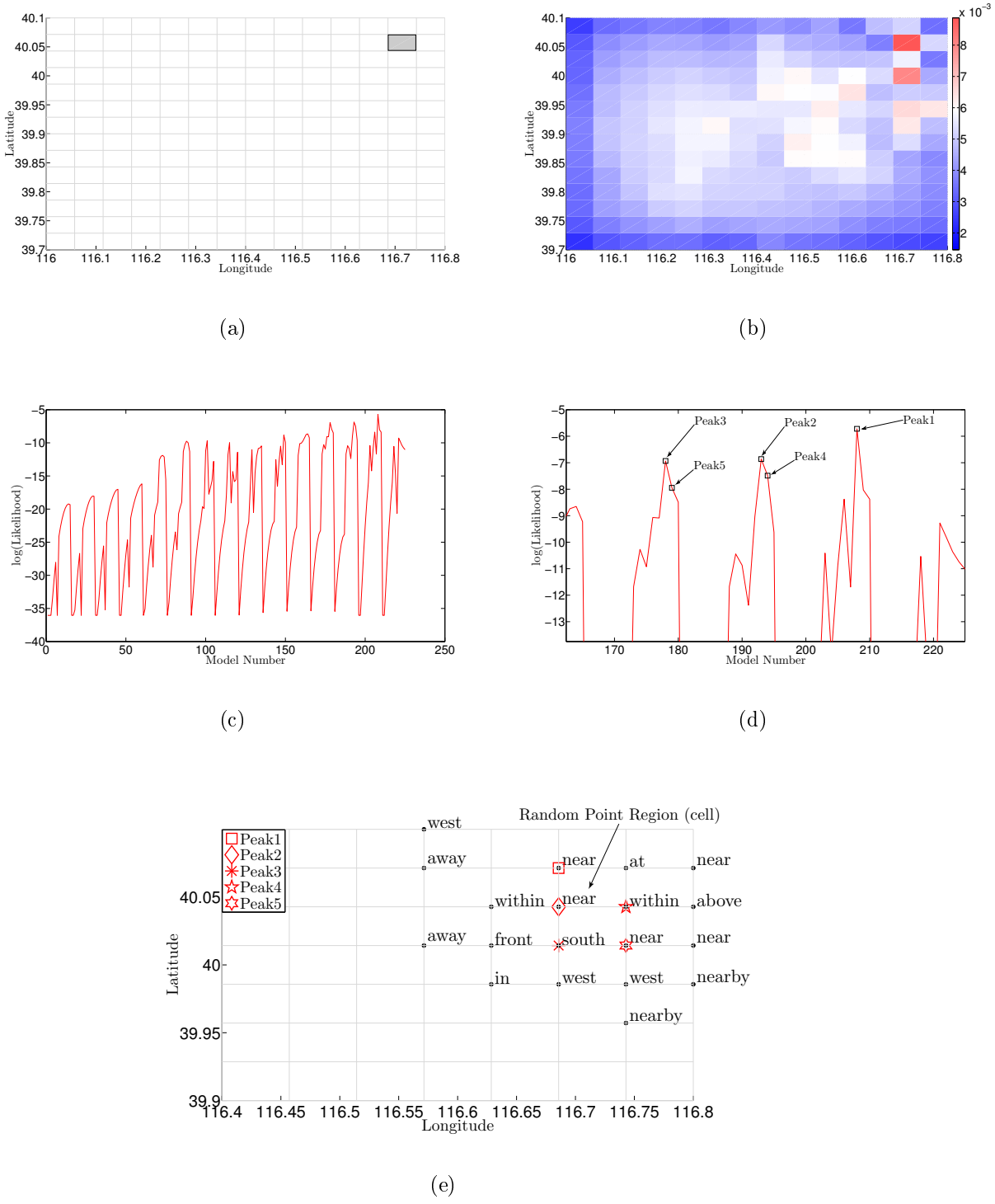
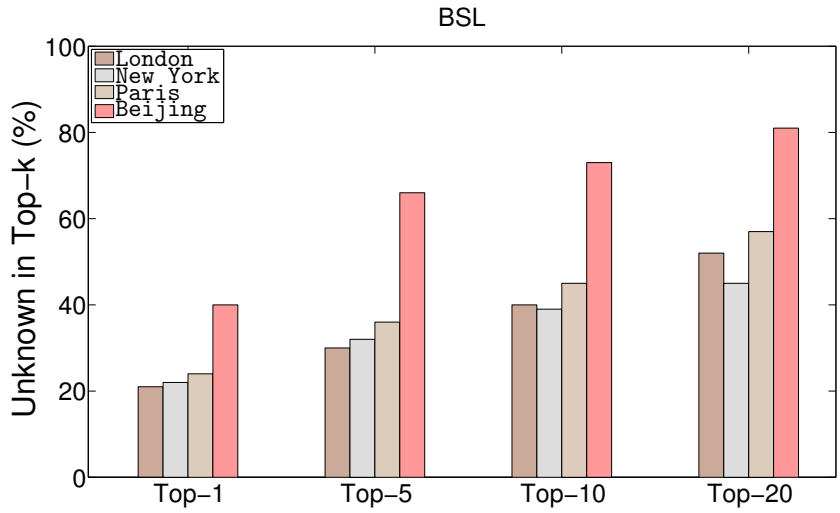
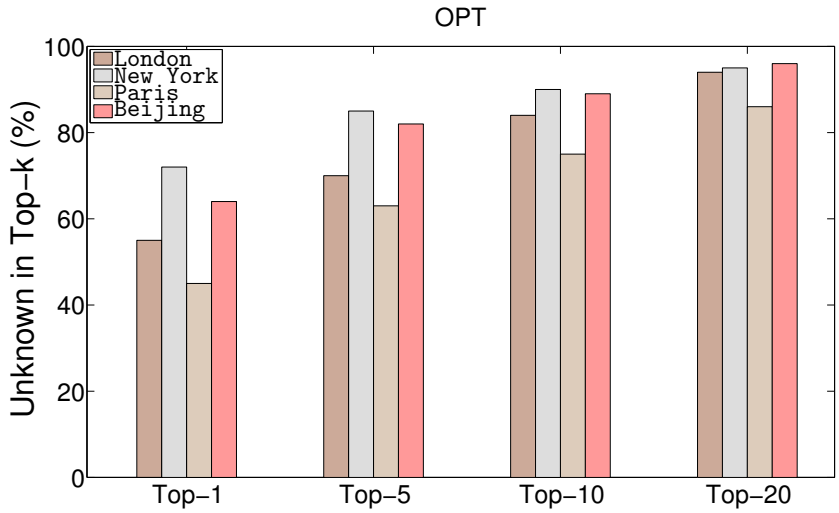


Figure 3.2: *A location estimation scenario: (a) the region in which a random point has been generated, (b) the probability of each region after a full run of Algorithm 3 is shown using heatmap colors, (c) the log-Likelihood of the random point's region as we traverse from one vertex (landmark) to the other, (d) an enlarged portion of (c) with the five highest likelihood peaks, and (e) the 20 best vertex-models, with the 5 best vertex-models emphasized in red (peaks in (d)).*

prediction accuracy improvement for the OPT model. In some cases (indicated in bold) the prediction accuracy improvement is equal to or greater than 30%.



(a)



(b)

Figure 3.3: *Location prediction accuracy. (a) Illustrates the prediction accuracy of the BSL model for K values $1, 5, 10, 20$ respectively. (b) Illustrates the prediction accuracy of the OPT model for K values $1, 5, 10, 20$ respectively.*

Finally, we measure the percentage of selected models $\hat{\mathcal{G}}'$ that are qualitatively correct, i.e., they reveal a true spatial relation between a vertex and a random point. Figure 3.4 shows the percentage correct models $\hat{\mathcal{G}}'$. The percentage of OPT model is quite higher than that of the BSL model and Table 3.2 shows this improvement in relative terms. In some cases (indicated in bold) the qualitative accuracy improvement is more than 10%.

To *visualize the actual models* and the respective probabilities they assign to

Table 3.1: *Prediction accuracy improvement when the optimized model (OPT) is used instead of the BSL model.*

	Improvement per Top-k case			
Dataset	$k = 1$	$k = 5$	$k = 10$	$k = 20$
London	+**34**%	+**40**%	+16%	+15%
New York	+**50**%	+**53**%	+**51**%	+**50**%
Paris	+21%	+27%	+**30**%	+29%
Beijing	+24%	+16%	+16%	+15%

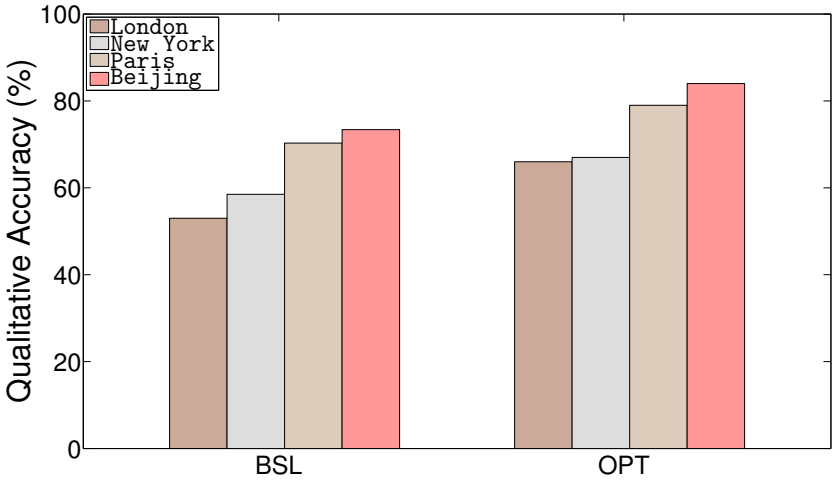
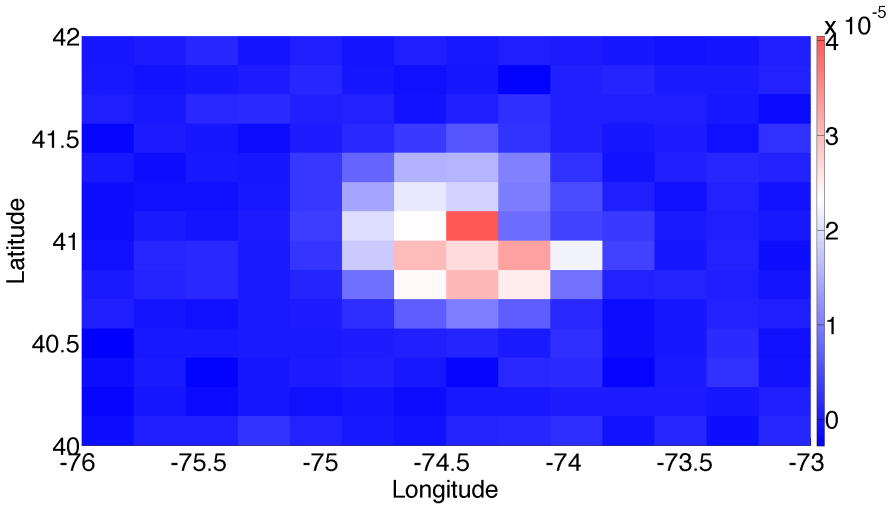


Figure 3.4: *Percentage of correct spatial relation models.*

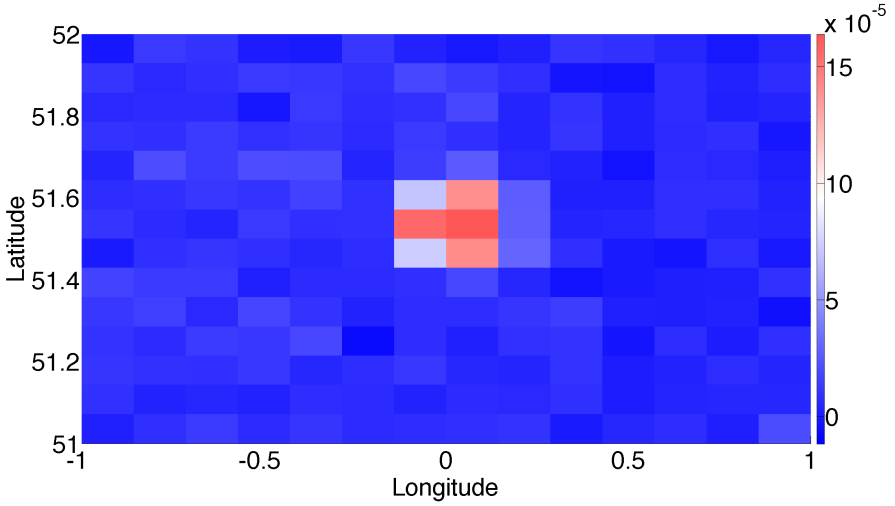
Table 3.2: *Qualitative accuracy improvement when the optimized model (OPT) is used instead of the BSL model.*

Dataset	Improvement
London	+8%
New York	+**11**%
Paris	+**13**%
Beijing	+9%

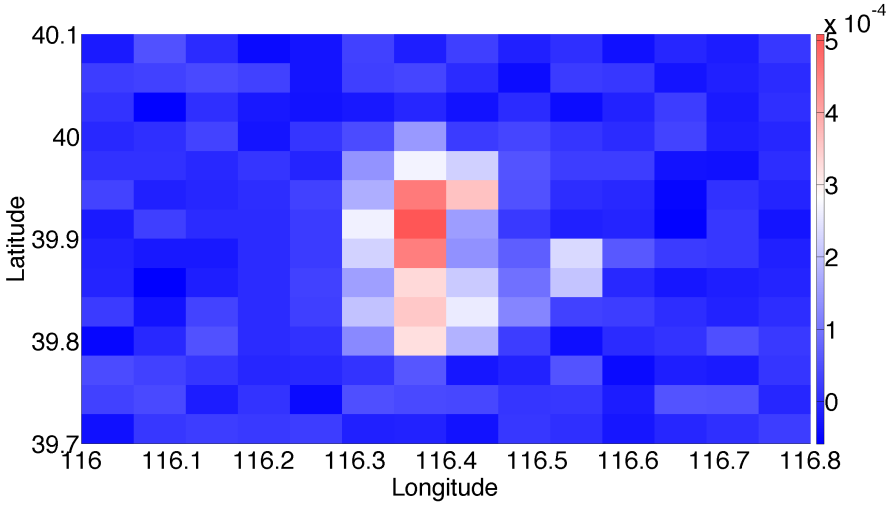
partitioned space, Figure 3.5 depicts three instances of spatial relations, with the center of the grid denoting a reference (landmark) point. The spatial extent of relations (a) *"Near"*, (b) *"At"* and (c) *"West"* when searching for an unknown point that is spatially related to the center of the grid. The examples have been derived from the New York, London, and Beijing datasets. The concentration of measures around qualitatively correct regions is a further indication for the correctness of our models.



(a) **Near** the grid center



(b) **At** the grid center



(c) **West** of grid center

Figure 3.5: *Spatial extension of spatial relationships - probabilities of specific spatial relationships (Near, At, West) relating vertices to the center grid cell.*

3.4.2 Location Estimation for Real-world Scenarios

The ultimate goal of this work is to train probabilistic models for spatial relationships so as to provide location estimation, e.g., finding the "best pita place in Greece", based on hints in the form of qualitative spatial relationships discovered in textual narratives. This is a potentially important method, as it provides a solution to the geocoding problem that exists for user-contributed data on the Web, i.e., there is a myriad of mentioned POIs whose coordinates do not exist in any gazetteer.

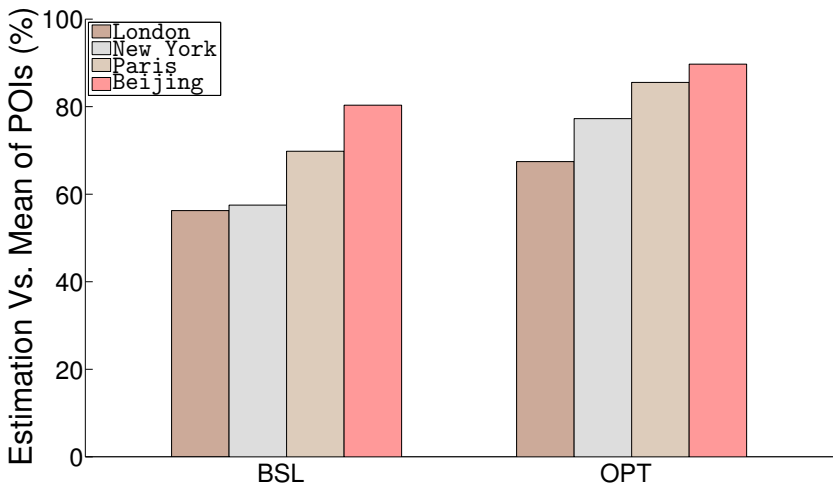
In addressing this challenge, we present extensive location estimation experiments on 3,000 real world scenarios extracted from all four datasets as they were presented in the Section 3.4.1. (about 800 real scenarios per region). We extract 3,000 POIs (considered as unknown) whose locations are given in (spatial) relation to known POIs. Note that these cases have *not* been used in the training phase. The experiments will also show the impact of the number of components per spatial relationship model on the quality of the location estimation outcome, e.g., the performance improvement when using the OPT instead of the BSL model. The spatial relation fusion procedure in the real-world scenarios is the same as presented in Algorithm 3, with the difference that the reference landmarks are real POIs (not grid vertices) extracted from textual narratives, and that we use the observed, in text, spatial relation model instead of the selected model $\hat{\mathcal{G}}'$.

3.4.2.1 Naïve Method

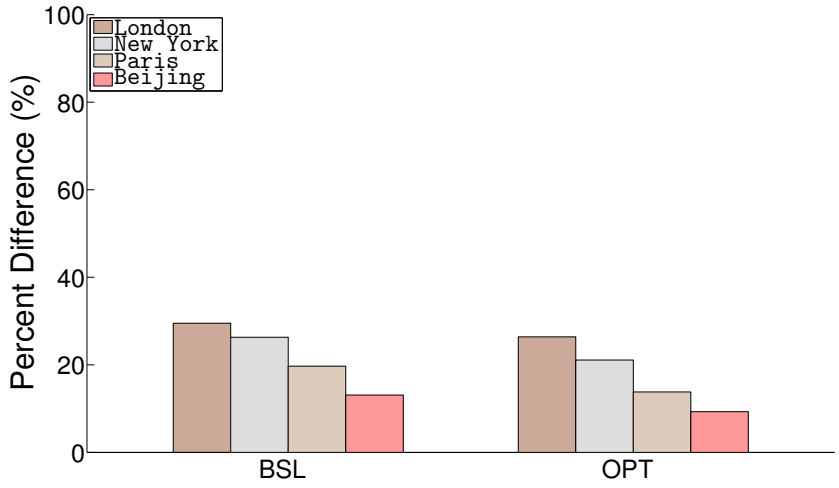
The first experiment estimates the location of an unknown POI using BSL and OPT models and *compares the result to the mean location of all referenced POIs*. The results for all four datasets are given in Figure 3.4.2.1. The one-component BSL model provides better location estimation (center of spatial probability density) when compared to to the mean location. It is closer to the actual POI's location in 56%, 57%, 70% and 80% of the cases, for the datasets of London, New York, Paris and Beijing, respectively. Using the OPT model improves the result further by 12%, 20%, 16% and 10% for each dataset, respectively. For the cases in which the mean location is closer to the unknown POI location than the center of spatial probability density, Figure 3.4.2.1 shows the actual differences in percent (of the distance of the mean location). The results show that while not outperforming the mean location for a small number of cases, the spatial probabilities in any event come close. Overall the results show that for scenarios with good data coverage such as Beijing (many spatial relationships extracted from texts), the spatial probabilities almost always ($> 90\%$) outperform a naïve method, and even if they do not, they produce almost identical results.

3.4.2.2 Location Estimates

Having established the validity of our approach, we want to measure the *distance of the estimated to the actual POI location*. Table 3.3 illustrates the percentage of location estimation scenarios for each of the defined distance buckets (0-2km, 2-4km, 4-6km, 6-8km. $> 8km$) for both methods and all four datasets. *The more results fall into the short distance buckets, the better the estimation for the specific case.* Again, an improvement of the result quality can be observed when contrasting the



(a)



(b)

Figure 3.6: *Location prediction accuracy. (a) Percentage of real scenarios - center of spatial probability density closer than the mean location of referenced POIs. (b) Difference in percentage of the distance of the mean location - mean location closer than the center of spatial probability density.*

BSL and OPT methods, as with the latter, the shorter distance percentage increases. Specifically, for the case of London and New York there is an increase (indicated in bold) in the first (0-2 km) and third (4-6km) bucket percentages, while for the Paris and Beijing datasets there is an increase in the first (0-2 km) and second (2-4km) distance buckets. This means that by using the OPT method, we obtain preciser location estimates.

Assuming a perfect method, all results would be in the first bucket. The case of Beijing comes close to this ambition as 29% and 35% (64% total) of the estimates

Table 3.3: *Prediction accuracy in terms of estimated distance from the unknown POI location.*

	Dataset							
	London		New York		Paris		Beijing	
Distance	*BSL*	*OPT*	*BSL*	*OPT*	*BSL*	*OPT*	*BSL*	*OPT*
0-2 *km*	8%	**11**%	14%	**24**%	10%	**12**%	21%	**29**%
2-4 *km*	36%	33%	43%	33%	30%	**33**%	22%	**35**%
4-6 *km*	30%	**31**%	21%	**30**%	37%	35%	39%	21%
6-8 *km*	17%	16%	15%	9%	16%	15%	13%	11%
> 8 *km*	9%	9%	7%	4%	7%	5%	5%	4%

are within $2km$ and $4km$ of the actual location, respectively.

3.4.2.3 Case Studies

To illustrate the impact of the number of observations on the result quality, we visualize four concrete location estimation scenarios (one for each dataset) by, in each case, progressively increasing the number of observations (spatial relationships). Figure 3.7 illustrates the aforementioned scenarios. Figures 3.7(a), (b) and (c) illustrate an unknown POI (red star) in the greater area of London, whose position is described in relation to known POIs (black stars) using a total number of 15 spatial relations. Figure 3.4.2.3 shows the contours of the spatial probability distribution when only using a randomly selected 50% of the observations, while Figure 3.4.2.3 shows the final distribution considering all spatial relations. Finally, Figure 3.4.2.3 is a closeup of Figure 3.4.2.3 with a GoogleMaps basemap overlay. In a similar fashion, Figure 3.7 shows the results for New York, Beijing and Paris, with a total number of 20, 70 and 200 spatial relations being used in each case, respectively. These results demonstrate the considerable prediction accuracy. Especially the estimates for Beijing (see Figures 3.4.2.3, 3.4.2.3, 3.4.2.3) and Paris (see Figures 3.4.2.3, 3.4.2.3, 3.4.2.3) *clearly pinpoint the unknown POI location.* What is further encouraging is that even for the cases of London (see Figures 3.4.2.3, 3.4.2.3, 3.4.2.3) and New York (see Figures 3.4.2.3, 3.4.2.3, 3.4.2.3), for which the number of relations is small, the proposed approach works reasonably well.

As expected, the prediction accuracy increases with the number of observations (models) considered. This is confirmed by the mass of the probability moving closer to the unknown POI location when increasing the number of observations from a randomly selected 50% (Figure 3.7 1^{st} column) to 100% (Figure 3.7 2^{nd} column). This effect is observed for all four cases. Table 3.4 shows the actual distances between the centers of the spatial probability distributions and the actual POI locations as we increase the percentage of spatial relations considered in our estimate.

The results show that as we increase the number of relations considered, we achieve more accurate estimates. The improvement is considerable for all cases, with Beijing and Paris benefitting most and achieving distances of $< 2km$ (indicated in bold in Table 3.4). Moreover, although the estimation quality (accuracy as well as precision) increases with the number of observations, nevertheless, even in the case of a small number of available observations, we can rely on the crowd as a data

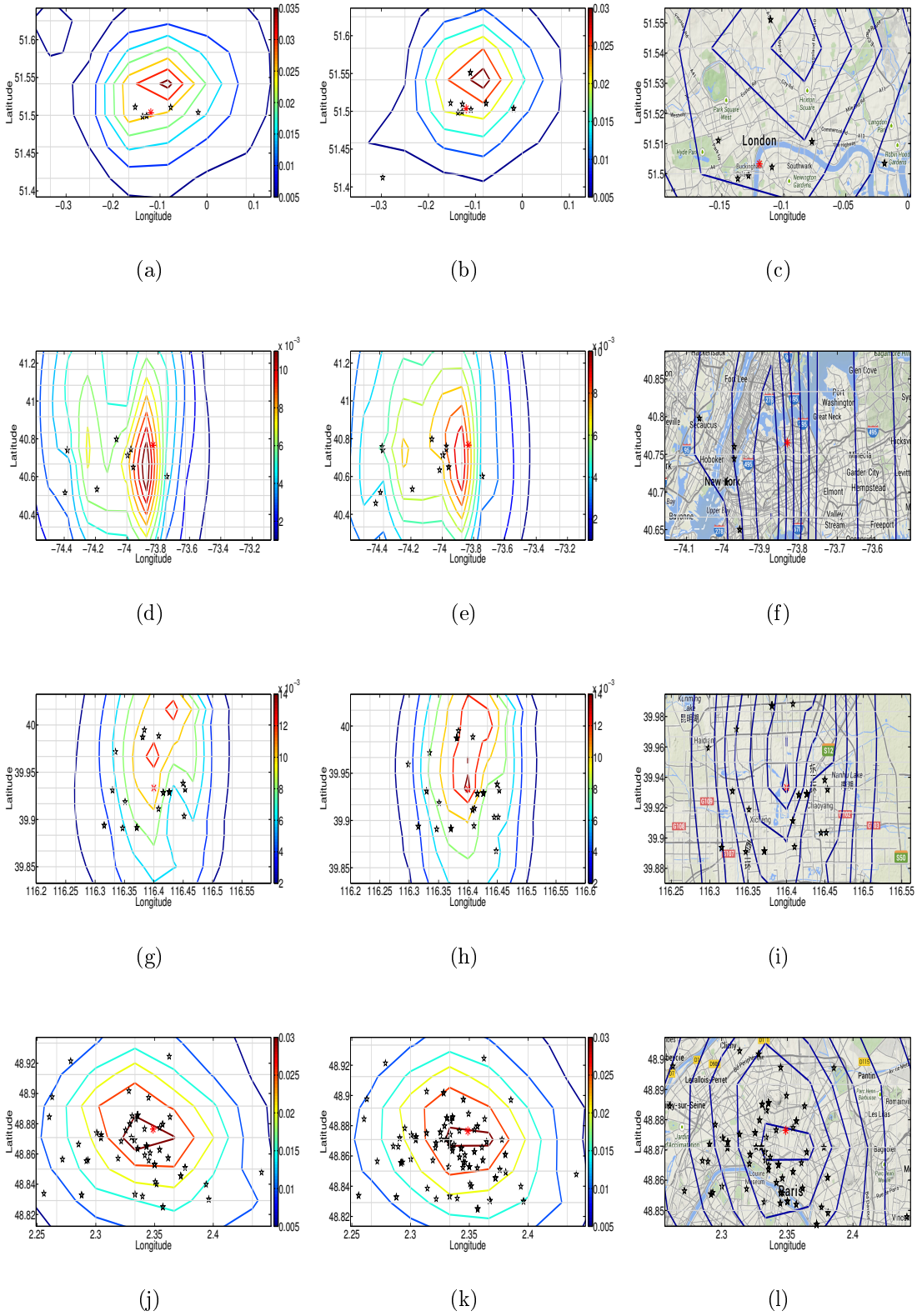


Figure 3.7: *Real world location estimation scenarios - Rows 1 to 4 are scenarios for London, New York, Beijing, and Paris - Columns 1-3 shows results for 50% , 100% and 100% (on Google Maps) of the observations (discovered relations) considered in the estimation.*

Table 3.4: *Distance between the center of the spatial probability distribution and the unknown POI.*

	Percentage of relations considered		
Dataset	10%	50%	100%
London	15.3km	7.9km	7.7km
New York	16.2km	11.9km	11.1km
Beijing	14.4km	8.6km	**1.2**km
Paris	8.7km	**1.6**km	**0.8**km

source for location estimation.

Overall, we can conclude that the proposed modeling using GMMs optimized by the greedy EM algorithm presented in Section 2.2.3.3 can efficiently handle the uncertainty introduced by user-contributed qualitative geospatial data. In combination with information extraction techniques, it provides us with the non-trivial means of textual narrative-based location estimation.

Chapter 4

Popular Path Computation

Directions and paths, as commonly provided by navigation systems, are usually derived considering absolute metrics, e.g., finding the shortest or the fastest path within an underlying road network. With the aid of Volunteered Geographic Information (VGI), i.e., geo-spatial information contained in user generated content, we aim at obtaining paths that do not only minimize distance but also lead through more popular areas. Based on the importance of landmarks in Geographic Information Science and in human cognition, we extract a certain kind of VGI, namely spatial relations that define *closeness* (*nearby*, *next to*) between pairs of *points of interest* (POIs), and quantify them following a probabilistic framework. Subsequently, using Bayesian inference we obtain a crowd-based *closeness* confidence score between pairs of PoIs. We apply this measure to the corresponding road network based on an altered cost function which does not exclusively rely on distance but also takes crowdsourced geo-spatial information into account. Finally, we propose two routing algorithms on the enriched road network. To evaluate our approach, we use Flickr photo data as a ground truth for popularity. Our experimental results – based on real world datasets – show that the paths computed w.r.t. our alternative cost function yield competitive solutions in terms of path length while also providing more "popular" paths, making routing easier and more informative for the user.

4.1 Preliminaries

User generated content has benefited many scientific disciplines by providing a wealth of new data. Technological progress, especially smartphones and GPS receivers, has facilitated contributing to the plethora of available information. OpenStreetMap constitutes the standard example and reference in the area of VGI. Authoring geo-spatial information typically implies coordinate-based, *quantitative data*. Contributing quantitative data requires specialized applications (often part of social media platforms) and/or specialized knowledge, as is the case with OpenStreetMap (OSM).

The broad mass of users contributing content, however, are much more comfortable using *qualitative information*. People typically do not use geo-coordinates to describe their spatial motion, for instance when traveling or roaming. Instead, they use qualitative information in the form of toponyms (landmarks) and spatial relationships ("near", "next to", "close by", etc.). Hence, there is an abundance of geo-spatial information (freely) available on the Internet, e.g., in travel blogs, largely

unused. In contrast to quantitative information, which is mathematically measurable, qualitative information is based on personal cognition. Therefore, accumulated and processed qualitative information may better represent human way of thinking.

This is of particular interest when considering the "routing problem" (equivalent to "path computation"). Traditional routing queries use directions from systems that only take inherent cost measure of the underlying road network into account, e.g., distance or travel time. In human interaction, such information is usually enhanced with qualitative information (e.g. "the street next to the church", "the bridge north of the Eiffel tower"). Combining traditional routing algorithms with crowdsourced geo-spatial references we aim to more properly represent human perception while keeping it mathematically measurable.

In [RW14], the authors analyze the important role of landmarks for the representation of geographic space in human mind, i.e., people tend to describe their position in space based on landmarks and relations between them. Based on this fact, in this work, we enrich a road network with information about spatial relations between pairs of Points of Interest (POI) extracted from user generated data (travelblog data). Using these relations, we obtain routes that are easier to interpret and follow, possibly rather resembling a route that a person would provide.

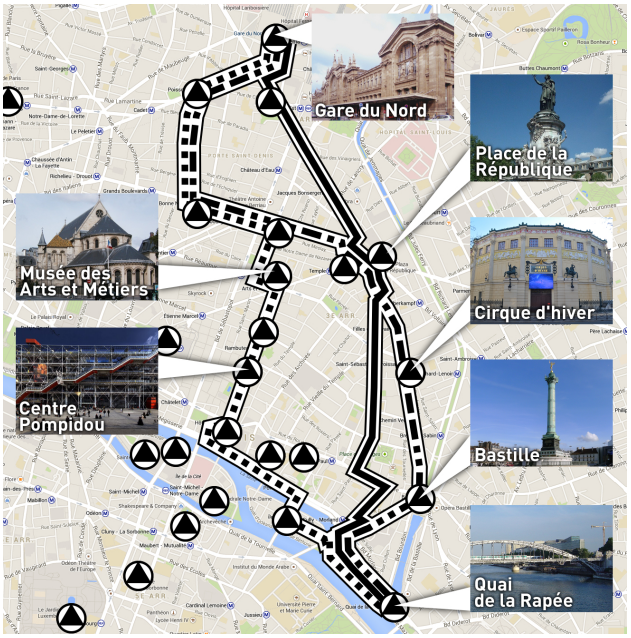


Figure 4.1: *Shortest (continuous) and alternative paths (dot dashed and dotted) alongside POIs in the city of Paris. This result is an output of some of the algorithms presented in this dissertation.*

As an example, consider the routing scenario in Figure 4.1 which is set in the city of Paris, France. The continuous line represents the conventional shortest path from starting point "Gare du Nord" to the target at "Quai de la Rapée" while the dot dashed and dotted lines represent alternative paths computed by the algorithms introduced in this paper. The triangles in this example denote touristic landmarks and sights. For instance, the dot dashed path on the bottom right passing recognizable locations such as "Place de la République", "Cirque d'hiver" and "la Bastille", as proposed by our algorithms, is considerably easier to describe and follow, and might yield more interesting sights for tourists than the shortest path.

The major challenge in this contribution is the extraction of crowdsourced geospatial information from textual data and the enrichment of an existing road network with this information. The enriched road network is subsequently used to provide paths between a given start and target that satisfy the claim of higher popularity (which is formally introduced in Section 4.3.3), while only incurring a minor additional spatial distance. In addition to this main application, we note that our techniques can furthermore be used to automatically provide interesting tourist routes in any place where information about POIs is available. The transition from textual information to routing in networks is not at all straightforward, therefore we employ and develop various methods from different angles of computing science. To summarize, our contributions are as follows:

- We first mine VGI from user generated texts, by employing Natural Language Processing (NLP) methods in order to determine spatial entities (POIs) and spatial relations between them (Chapter 2).
- Due to the inherent uncertainty of crowdsourced data, we employ probability distributions to quantitatively model spatial relations mined from the text (Chapter 2).
- We propose a Bayesian inference-based transition from the modeled spatial relations to spatial *closeness* confidence measurements according to the crowd (Section 4.6.1.6).
- We define a new cost criterion which is used to enrich an underlying road network with the aforementioned confidence measurements based on the Djikstra shortest path algorithm (Section 4.3.3.2).
- We optimize the Dikstra based enrichment approach with a skyline-based road network enrichment approach (Section 4.3.3.4).
- Finally, we propose two algorithms which use the enriched road network to compute actual paths (see Section 4.4).

4.2 Related Work

Research areas relevant to this work include: (*i*) qualitative routing and (*ii*) mining of semantic information from moving object trajectories and trajectory enrichment with extracted semantic information. In what follows, we discuss previous work in both of these areas.

While finding shortest paths in road networks is a thoroughly explored research area, qualitative routing has hardly been explored. Nevertheless, providing meaningful routing directions in road networks is a research topic of great importance. In various real world scenarios, the shortest path may not be the ideal choice for providing directions in written or spoken form, for instance when in an unfamiliar neighborhood, or in cases of emergency. Rather, it is often more preferable to offer "simple" directions that are easy to memorize, explain, understand and follow. However, there exist cases where the simplest route is considerably longer than the shortest. The authors in [SB13] and [WR11] try to tackle the problem of efficient routing by using cost functions that trade off between minimizing the length of a

provided path while also minimizing the number of turns on the provided path. The major shortcoming of these approaches is that they focus almost exclusively on road network data without taking into account any kind of qualitative information, i.e., information coming from the user. Opposed to that, we try to approach the problem of efficient routing by integrating spatial knowledge coming from the crowd thus enriching an actual road network.

The discovery of semantic places through the analysis of raw trajectory data has been investigated thorougly over the course of the last years. The authors in [LCC12], [YCP+11], [PBKA08] and [And13], provide solutions for the semantic place recognition problem and categorize the extracted POIs into pre-defined types. Moreover, the concept of "semantic behavior" has recently been introduced. This refers to the use of semantic abstractions of the raw mobility data, including not only geometric patterns but also knowledge extracted jointly from the mobility data as well as the underlying geographic and application domains in order to understand the actual behaviour of moving users. Several approaches like [ABK+07], [PSR+13], [SP11], [YCP+13], [SP11] and [YSC+10] have been introduced the last decade. The core contribution of these articles lies in the development of a semantic approach that progressively transforms the raw mobility data into semantic trajectories enriched with POIs, segmentations and annotations. Finally, a recent work, [FSSR13], can extract and transform the aforementioned semantic information into a text description in the form of a diary. The major drawback of these approaches is that they do not intergrate the extracted semantic information into the road network. Instead, they use the extracted information only on specific trajectories. In our contribution, we analyze crowdsourced data in order to extract semantic spatial information and intergrate it into an actual road network. This will enable us to provide routes that are near-optimal w.r.t. distance while spatially more popular according to the crowd.

4.3 Contribution

This section highlights our approach on qualitative data extraction from texts and presents a probabilistic approach for representing spatial relationships based on distance and orientation features. Key ingredients of our approach are NLP methods for information extraction from texts and algorithms that train probabilistic models, which are required due to the inherent uncertainty of crowdsourced data. Our discussion below includes a short description of NLP tools we use to extract spatial relations between POIs, the features we used to model spatial relations as probability distributions, and a short analysis of the modeling approach used in [SPK13]. These models are necessary to assess the quality of spatial relations extracted from text which will be used in Section 4.3.3.2 for the enrichment of the underlying road network.

4.3.1 Spatial Relation Extraction from Texts

Here we follow the same path as described in Chapter 2. We choose travel blogs as a rich source for (crowdsourced) geo-spatial data. This selection is based on the fact that people tend to describe their experiences in relation to their trips and places they have visited, which results in "spatial" narratives. To gather such data, we

use classical Web crawling techniques and compile a database consisting of 250,000 texts, obtained from 20 travel blogs.

Obtaining qualitative spatial relations from text involves the detection of (*i*) POIs (or toponyms) and (*ii*) spatial relationships linking the POIs. The employed approach involves geoparsing, i.e., the detection of candidate phrases, and geocoding, i.e., linking the phrases to actual coordinate information.

For the relation extraction task we follow the approach used in [SPKS15] where a Natural Language Processing Toolkit (NLTK) (cf. [LB02]) based spatial relation extraction approach is presented. NLTK is a leading platform for analyzing raw natural language data. The search for spatial relations in texts results into triplets of the form (P_i, R^k, P_j), where P_i and P_j are named entities (landmarks) and R^k is the spatial relation that intervenes between P_i and P_j. Following this path, we managed to extract 500,000 POIs from the aforementioned travel blog text corpus. For the geocoding of the POIs, we rely on the GeoNames[1] geographical gazetteer data, which contains over ten million POI names worldwide and their coordinates. This procedure associates (whenever possible) POIs found in the travel blogs with geo coordinates. Using the GeoNames gazetteer we were able to geocode about 480,000 out of the 500,000 extracted POIs and to end up with about 600,000 triplets of the form (P_i, R^k, P_j) worldwide.

For our experiments we want to focus on regions with high triplet density in order the get meaningful results. Therefore, we focus on the cities of Paris and New York. The triplets we extracted for these two cities define a what we call *Spatial Relationship Graph*, i.e., a spatial graph in which nodes represent POIs and edges are spatial relationships between them. Let us point out that for the scope of this work, i.e., a combination of short and enriched routes, we only consider distance and topological relations that denote closeness (near, close, next to, at, in etc). The use of relations that denote direction, e.g., north, south, east etc., or remoteness, e.g., away from, far etc., is an open direction for future work.

4.3.2 Modeling Spatial Relations

4.3.2.1 Feature Extraction

In order to train probabilistic models, we need informative features. We model each spatial relation in terms of *distance* and *orientation* as presented in [SPK13]. Therefore, we extract occurrences of a spatial relation (such as "near") from travel blogs. For each occurrence, we create a two-dimensional spatial feature vector $D = (D_d, D_o)^\intercal$ where D_d denotes the distance and D_o denotes the orientation between P_i and P_j. Specifically, assuming a projected (Cartesian) coordinate system, the distance between two POIs P_i and P_j is computed as the Euclidean metric between the two respective coordinates. The orientation is established as the counterclockwise rotation of the x-axis, centered at point P_j, to point P_i. This way, we end up with a set of two-dimensional feature vectors $\mathcal{D}_{\text{rel}} = \{D_1, D_2, \ldots, D_n\}$ for each spatial relation. We will use the set of two-dimensional feature vectors in order to train a probabilistic model for each spatial relation.

[1] http://www.geonames.org/

4.3.2.2 Probabilistic Modeling

As described in [SPK13] and Chapter 2, by using a set of two-dimensional feature vectors for each spatial relation such as "near" or "into", we can train Gaussian Mixture Models (GMMs), which have been extensively used in many classification and general machine learning problems ([Bis06]).

In general, a GMM is a weighted sum of M-component Gaussian densities as $p(d|\lambda) = \sum_{i=1}^{M} w_i g(d; \mu_i, \Sigma_i)$ where d is a l-dimensional data vector (in our case $l = 2$), w_i are the mixture weights, and $g(d; \mu_i, \Sigma_i)$ is a Gaussian density function with mean vector $\mu_i \in \mathbb{R}^l$ and covariance matrix $\Sigma_i \in \mathbb{R}^{l \times l}$. To fully characterize the probability density function $p(d|\lambda)$, one requires the mean vectors, the covariance matrices and the mixture weights. These parameters are collectively represented in $\lambda = \{w_i, \mu_i, \Sigma_i\}$ for $i = 1, \ldots, M$.

Let $\mathcal{R} = \{R^1, \ldots, R^n\}$ denote the set of all spatial relations that we take into account. In our setting, each relation R^k is modeled under a probabilistic framework by a 2-dimensional GMM, trained on each relation's set of two-dimensional feature vectors $\mathcal{D}_{\text{rel}}$. For the parameter estimation of each Gaussian component of each GMM, we use Expectation Maximization (EM) ([DLR77]). EM enables us to update the parameters of a given M-component mixture with respect to a feature vector set $\mathcal{D}_{\text{rel}} = \{D_1, \ldots, D_m\}$ with $1 \leq j \leq m$ and all $D_j \in \mathbb{R}^l$, such that the log-likelihood $\mathcal{L} = \sum_{j=1}^{m} \log(p(D_j|\lambda))$ increases with each re-estimation step, i.e., EM re-estimates model parameters λ until convergence. Further details on modeling spatial relations under a probabilistic framework are given in [SPK13].

This procedure results in a trained GMM of the form $p_k(D|\lambda)$, for each spatial relation $R^k, 1 \leq k \leq n$. Given a distance and orientation vector, we can use this model to estimate the probability that a particular relation exists. Based on this information, by bayesian inference we derive a closeness score for pairs of POIs. This procedure is described in the next section.

4.3.3 Road Network Enrichment

In this section, we describe our approach to enrich an actual road network with crowdsourced geo-spatial information. Our discussion below includes a description of how we transform a *Spatial Relationship Graph*, as presented in Section 4.3.1, into a weighted graph, and how we use the edge weights of the weighted graph in order to modify the edge costs of a real road network.

4.3.3.1 From Relationship to Weighted Graphs

As presented in Section 4.6.1.3, the spatial relation extraction procedure results in a relationship graph between POIs. A simple example of such a graph is shown in Figure 4.2. In general, let $\mathcal{P} = \{P_1, \ldots, P_m\}$ denote the set of nodes representing the POIs, and let $\mathcal{R} = \{R^1, \ldots, R^n\}$ denote the pre-defined set of spatial closeness relations, represented by spatial NLP expressions like "next to" or "close by".

Furthermore, let $R_{i,j} \subseteq \mathcal{R}$ denote the set of relations extracted from the text between two distinct nodes P_i and P_j. Note that R^k denotes an abstract relation, while $R_{i,j}$ denotes a set of occurrences of relations between a pair of nodes. Let $D_{i,j}$ denote the spatial feature vector (distance and orientation), between two distinct POIs P_i and P_j (as presented in Section 4.3.2.1). Finally, let $\mathcal{D} := \bigcup_{i \neq j \wedge R_{i,j} \neq \emptyset} D_{i,j}$

denote the set of all spatial feature vectors between all pairs of POIs which have non-empty sets of relations.

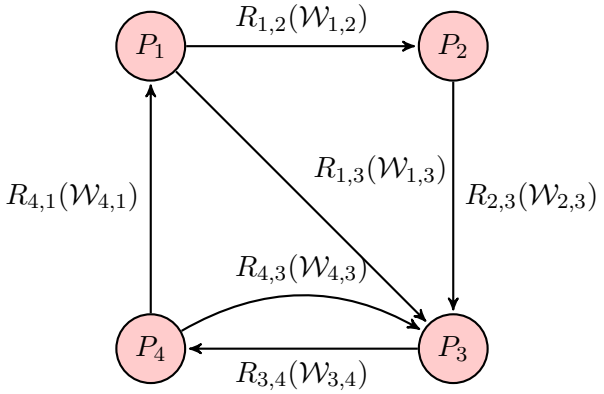


Figure 4.2: *Simple relationship graph. Nodes represent POIs and each edge represents the set of relations $R_{i,j}$through which its adjacent nodes P_i and P_j are connected. Each of these sets is mapped onto the closeness score $\mathcal{W}_{i,j}$, turning the relationship into a weighted graph.*

We want to estimate the posterior probability of a class $R^k \in R_{i,j}$ based on the spatial feature data $D_{i,j}$ between two POIs P_i and P_j. This is given by Equation 4.1. Here, $p(D_{i,j}|R^k)$ denotes the likelihood of $D_{i,j}$ given relation R^k based on the trained GMM (presented as $p(D|\lambda)$ Section 4.3.2.2), while $P(R^k)$ denotes the prior probability of relation R^k given only the observed relations $R_{i,j}$.

$$P(R^k|D_{i,j}) = \frac{p(D_{i,j}|R^k)P(R^k)}{\sum_{l=1}^{n} p(D_{i,j}|R^l)P(R^l)} \tag{4.1}$$

In a traditional classification problem the spatial relation R^k between a pair of POIs would be classified to the spatial relation model with the highest posterior. In contrast to this approach, we consider each posterior probability $P(R^k|D_{i,j})$ as a measure of confidence of the existence of relation R^k between P_i and P_j. Remember that all the relations we consider reflect terms of spatial closeness. We combine all these posteriors into one measure which we refer to as *closeness score* $\mathcal{W}_{i,j}$ of the pair of POIs P_i and P_j, defined in Equation 4.2.

$$\mathcal{W}_{i,j} = \frac{1}{|\mathcal{R}|} \cdot \sum_{i=1}^{|R_{i,j}|} \frac{P(R^k|D_{i,j})}{\max_k\{P(R^k|\mathcal{D})\}} \tag{4.2}$$

Here, we sum all the posteriors $P(R^k|D_{i,j})$ normalized by the maximum posterior of each relation in the relationship graph and we normalize the summation by the total number of spatial relations in the relationship graph. This is done for all pairs P_i, P_j where $R_{i,j} \neq \emptyset$. We refer to these pairs as *close* since at least one of our relations reflecting closeness exists. As is illustrated in Figure 4.2, assigning the respective weights $\mathcal{W}_{i,j}$ to the edges of the relationship graph, we obtain a weighted graph. Note that $\mathcal{W}_{i,j} \in [0, 1]$ but typically $0 < \mathcal{W}_{i,j} \ll 1$. In Section 5.2.4 the influence of $\mathcal{W}_{i,j}$ on the results is examined, in particular, different scalings are tested. In this weighted relationship graph, denoted by H^*, there exists a vertex for

each POI and an edge (P_i, P_j) (equipped with weights $\mathcal{W}_{i,j}$ and Euclidean distances d_{ij}) for each pair of POIs P_i, P_j that are close in the above sense ($R_{i,j} \neq \emptyset$).

4.3.3.2 From Weighted Graphs to Road Network Enrichment

Now that we have extracted and statistically condensed the crowdsourced data into a closeness score, we need to apply the obtained closeness scores to the underlying network. We have investigated several strategies and have decided upon a compromise between simplicity and effectiveness. We will present two road network enrichment approaches and we propose two algorithms on routing with enriched graphs. The first enrichment approach, also analyzed in our previous work in [**?**], is based on Djikstra shortest path computation while the second is based on Skyline path computation.

Initially, let $G = (V, E, d)$ denote the graph representing the underlying road network, i.e., the vertices $v \in V$ correspond to crossroads, dead ends, etc., the edges $e \in E = V \times V$ represent roads connecting vertices. Furthermore, let $d : E \to \mathbb{R}_0^+$ denote the function which maps every edge onto its distance. We assume that $\mathcal{P} \subseteq V$, i.e., each POI is also a vertex in the graph. This is only a minor constraint since we can easily map each POI to the nearest node of the graph or introduce pseudo-nodes. Our two enrichment methods are described below.

4.3.3.3 Djikstra Shortest Path Approach

For each pair of spatially connected POIs, P_i, P_j, we compute the shortest path connecting P_i and P_j in G, which we denote by $r(i, j)$. We then define a new cost function $c : E \to \mathbb{R}_0^+$ which modifies the previous cost $d(e)$ of an edge as follows:

$$c(e) = d(e) \cdot \prod_{e \in r(i,j)} (1 - \alpha \mathcal{W}_{i,j}) \tag{4.3}$$

where $e \in r(i, j)$ iff e is an edge within the shortest path from P_i to P_j and where $\alpha \in [0, 1]$ is a weight scaling factor to control the balance between the spatial distance $d(e)$ and the modification caused by the closeness score $\mathcal{W}_{i,j}$. In the case of $\alpha = 0$, we obtain the unadapted edge weight $c(e) = d(e)$. Summarizing, the more shortest paths between POI pairs run through e, the lower its adjusted cost $c(e)$. The reason for enriching the shortest paths is that they represent the most intuitive connections between any two points in a road network.

We now define the *enriched graph* $G^* = (V, E, c)$. It consists of the original vertices and edges and is equipped with the new cost function which implies the reweighting of edges. Any path computation algorithm in G^*(e.g. a Dijkstra search) therefore favors edges which are part of shortest paths between POIs which are close according to the crowd. When computing the cost of a path on G^*, as before, we sum the respective edge weights which now differ from the original edge weights (due to the altered cost function). We refer to this procedure of incorporating the crowdsourced information and the respective graph as *D-enrich*.

4.3.3.4 Path Skyline Approach

One shortcoming of *D-enrich* is the assumption that the crowd unanimously favors exactly one path to connect a pair of POIs P_i and P_j, namely the shortest path.

Especially in multicriteria networks which comprise of a set of cost criteria, e.g., travel time, energy consumption, road tolls, optimality is usually defined as a personal trade-off between the given criteria. For example: How much additional time has to be spent to avoid a toll road? However, defining this trade-off numerically as a vector of preferences is not reasonable, and even if it would be, finding the personally preferred trade-offs for all users is in general not possible. Therefore, the best practice is to present a set of alternative paths to the user. The most established and very comprehensive set of alternative paths is the so-called path skyline [**?**]. This set contains all paths which are non-dominated in the following sense: The cost vector u dominates a cost vector v, denoted $u \prec_{\text{dom}} v$, if u has a smaller cost value than v in at least one dimension i and v does not have a smaller cost value than u in any dimension j. Hence, the path skyline comprises all path which are optimal under some monotone combination function of the cost criteria. Hence, the path skyline contains all optimal paths for all possible trade-offs between the cost criteria.

To enrich our road network, we compute the path skyline (w.r.t. distance and travel time) as proposed in [SJSK] between each pair of spatially connected POIs P_i and P_j in G, denoted by $s(i,j)$. Although the paths contained in $s(i,j)$ differ from one another, they often share some edges. Simply following each path for enrichment might unnecessarily favor edges contained in many skyline paths. Therefore, we adjust the weights of edges independent of the number of skyline paths in which they occur. Let $S_{i,j} \subset E$ denote the set of all distinct edges which are part of at least one skyline path from P_i to P_j. Analogously to *D-enrich*, we define the cost function $c : E \to \mathbb{R}_0^+$ to modify the original cost $d(e)$ of an edge, as before. While the adjusted cost function is the same as before (see Equation 4.3), the set of edges with adjusted costs is a superset, i.e., $S_{i,j} \supseteq r(i,j)$.

We now define the *enriched graph* $G^{**} = (V, E, c)$. It consists of the original vertices and edges equipped with the altered cost function reflecting a re-weighting of edges contained in skyline paths. Any path computation algorithm in G^{**}(e.g. a Dijkstra search) therefore favors edges which are part of the Skyline paths between POIs which are close according to the crowd. We refer to this procedure of incorporating the crowdsourced information and the respective graph as *S-enrich*.

4.3.3.5 Influence of Adjusted Costs

In order to measure the influence of the adjusted cost values along a computed path $p = (e_1, \ldots, e_r)$ on an enriched graph (G^*or G^{**}), we introduce the *enrichment ratio* (ER) function *er*.

$$er(p) = \frac{1}{d(p)} \sum_{i=1}^{r} c(e_i) \tag{4.4}$$

Here, $d(\cdot)$ and $c(\cdot)$ are as in the previous two sections. By normalizing with the total length of the path, we are able to compare the spatial connectivity of paths independent of length as well as start and target nodes. Here, a lower ratio implies higher closeness score values along the edges of the path. If none of the edges of a path is part of any shortest or skyline path between POIs, its enrichment ratio is 1, while the (highly unlikely) optimal enrichment ratio is 0. On the enriched graphs G^* and G^{**} we may now define our path computation algorithms.

4.4 Path Computation on Enriched Graphs

Now that we have a measure quantifying the enrichment of a path, we investigate the effect of *D-enrich* and *S-enrich* on the actual path computation. For this purpose, we present two approaches which make use of the enriched network and the weighted relationship graph H^* (Section 4.6.1.6). Continuing, they are compared to the conventional shortest paths within the original graph, as obtained with Dijkstra's algorithm, which we denote by Dij-G.

Note that for the evaluation procedure, all paths in this paper are computed by Dijkstra's algorithm because our main focus is not the routing itself but the incorporation of textual information into existing road networks. If desired, speed-up techniques, such as preprocessing steps and/or other search algorithms, could easily be employed.

Our first approach, given start and target nodes, it executes a Dijkstra search in the enriched road network graph G^* or G^{**} w.r.t. the adjusted cost function. Depending on the enrichment used, *D-enrich* or *S-enrich*, we refer to the first algorithm as Dij-G*or Dij-G**, respectively.

Our second approach, uses the enriched road network graphs G^* or G^{**} as well as the weighted relationship graph H^*. Given start and target nodes within the enriched graph (G^*or G^{**}), entry and exit nodes within H^* are determined. Subsequently, we route within H^*, i.e., from POI to POI, again using Dijkstra's algorithm. Depending on the enrichment used, *D-enrich* or *S-enrich*, we refer to the second approach we want to present as Dij-H*or Dij-H**, respectively. Note that in both cases we use the same graph H^*, but we refer to the *S-enrich* case as Dij-H** in order to differentiate the two methods.

All our approaches return paths connecting start and target. But while Dij-G computes the shortest path in the original graph G, all the approaches compute the shortest paths in the enriched graphs w.r.t. the adjusted cost function c. By construction of c, it favors edges which are part of the Dijkstra shortest paths or the skyline paths, between close POIs. Dij-H* and Dij-H** in contrast, do not only favor these edges, but are restricted to them. Having found entry and exit nodes within H^*, Dij-H* and Dij-H** hop from POI to POI in direction of the target. Hence, Dij-G, Dij-G*, Dij-G**, Dij-H*, Dij-H** in that order, represent an increasing binding to the extracted relations. Dij-G is not bound to the relations at all, while Dij-G* and Dij-G** (by the adjusted cost function) favors "relation-edges", and Dij-H* and Dij-H** are strictly bound to the relations and the graph formed by them.

Let us formalize Dij-H* (Dij-H** can be formalized in the same way). Given start and target node in G^* (or G^{**} for the Dij-H** case), it first determines the so-called entry and exit nodes to and from H^*. However, to exclude POIs which would imply a significant detour, we restrict the set of valid POIs, i.e., we restrict the search to a subgraph of H^*, denoted as h^*. Figure 4.3 illustrates our computationally inexpensive implementation of a query ellipse that allows for some deviation in the middle of the path as well as for minor initial and final detours.

The pseudo-code for the second approach is given in Algorithm 4. Here, we present only the Dij-H*case, since Dij-H** works in the same way by utilizing the G^{**} graph. After selecting the valid set of POIs (Step 2), entry and exit nodes to and from H^* are determined, i.e., the closest POIs to start and target node, respectively (Steps 4 and 5).

ALGORITHM 4: Dij-H*

Input: Enriched Graph G^*, Spatial Relationship Graph H^*, start s, target t

Output: Path p between s and t

```
1  begin
2      h* ← subgraph of H* in bounding ellipse
3      p ← empty path
4      P_entry ← select POI P ∈ h* closest to s
5      P_exit ← select POI P ∈ h* closest to t
6      p_h ← Dijkstra(h*, P_entry, P_exit)
7      predecessor ← s
8      foreach POI P on path p_h do
9          v ← select node v ∈ G* representing P
10         p.Append(Dijkstra(G*, predecessor, v))
11         predecessor ← v
12     end
13     p.Append(Dijkstra(G*, last, t))
14     return p
15 end
```

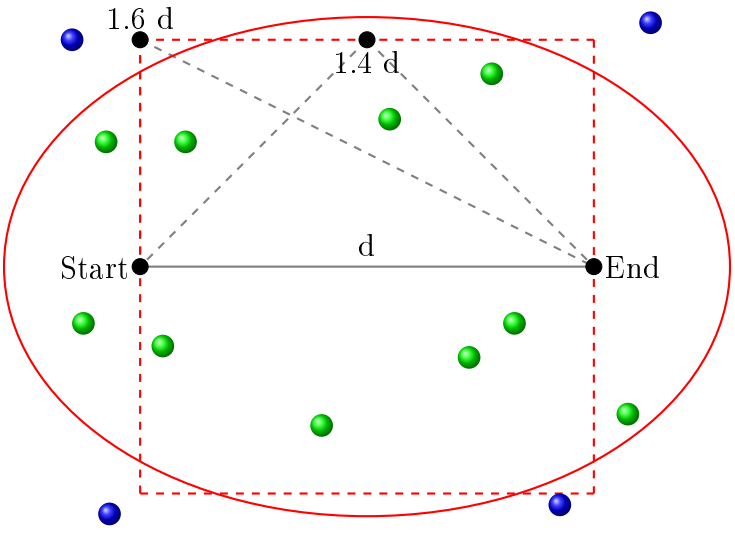


Figure 4.3: *Restriction of relationship graph H^* to a subgraph h^*, in order to avoid implausible detours. The green dots represent POIs, i.e., nodes of H^* which are also in h^*, the blue ones are left out.*

Entry and exist nodes connect the road network G^* to the relationship graph H^*. Subsequently, the shortest path in h^* from entry to exit node is computed using Dijkstra's algorithm w.r.t. the Euclidean distance (Step 5). Note that a shortest path within H^* is a sequence of POIs. We therefore map this sequence onto G^* by computing the shortest paths between the consecutive pairs of POIs in G^* w.r.t. the adjusted cost function (Step 8). Also, we compute the shortest paths in G^* from start to entry node and exit to target node. Concatenating these paths (start to

entry, POI to POI, exit to target), we return a full path.

4.5 Experimentation

In this section, we want to investigate the effect and impact of the network enrichment. We compare the results of the conventional Dijkstra search, Dij-G, to the results of Dij-G*and Dij-H*, which use the Djikstra shortest path enriched (*D-enrich*) graph G^*, and the results of Dij-G**and Dij-H**, which use the skyline path enriched (*S-enrich*) graph G^{**}. All approaches are evaluated on real world datasets. Besides comparing the computed path w.r.t. their enrichment ratio (ER) and length (as presented in Section 4.3.3.2), we introduce a measure of popularity based on Flickr data, which is explained in the following section. All the text processing parts were implemented in Python while modeling parts were implemented in Matlab. Network enrichment and path computation tasks were conducted using the Java-based MARiO Framework [GKRS] on an Intel(R) Core(TM) i7-3770 CPU at 3.40GHz and 32 GB RAM running Linux (64 bit).

4.5.1 Enrichment Ratio, Distance and Popularity Evaluation

Our experiments are set in two cities, Paris and New York. These regions have comparatively high density of spatial relations, Flickr photo data, and OSM data, which accounts for an exact representation of the road networks. As mentioned before, we compare the output of Dij-G, Dij-G*, Dij-H*, Dij-G**and Dij-H** w.r.t. to the paths they return, more precisely, w.r.t. ER and length of these paths. Since ER is a measure introduced in this paper, we use Flickr data as an independent ground truth. We are aware that to cognitive aspects (like the importance of sights or the value of landmarks) there is no absolute truth. However, in order to be able to draw comparisons, we presume that if the dataset is large enough, the bias can be neglected. We use a geotagged Flickr photo dataset, provided by the authors in [MSWHD], to assign a number of photos to each vertex of the underlying road network. The number of Flickr photos assigned to each vertex is referred to *popularity*. In our settings, every photo which is within the 20-meter radius of a vertex, contributes to the popularity of that vertex. The popularity of a path is computed by summation of all popularity values along this path.

The sizes of the weighted relationship graphs H^*, road network and Flickr photo data for both cities are shown in Table 4.1. Regarding the weighted relationship graphs, we provide the number of unique POI pairs extracted from the travel blog corpus and the number of spatial (closeness) relations extracted between them, as was presented in Section 4.6.1.3. Regarding Flickr data, we provide the total number of geotagged photos in each city and the maximum number of photos assigned to one vertex of the road network. Finally, regarding the road network, we provide the total number of edges and vertices. Note that although the datasets differ in terms of density (w.r.t. to relations and Flickr photos), our algorithms provide similar results.

We present two experimental settings: In Setting (i) we examine the influence of different scalings of the closeness score $\mathcal{W}_{i,j}$ in terms of enrichment ratio, path length increase (distance) and popularity. Setting (ii) investigates the influence of the path length, i.e., the distance between start and target is varied, again in terms

Table 4.1: *Statistics for the weighted relationship graphs, Flickr datsets and road networks of Paris and New York respectively.*

	Rel. Graph (H^*)		Flickr		Road Net. (G)	
Data	# Pairs	# Relations	# Photos	# Photos per Vertex	# Vertices	# Edges
Paris	400	2000	$400K$	100	$550K$	$300K$
NY	300	1500	$90K$	200	$220K$	$120K$

of enrichment ratio, path length increase (distance) and popularity. In both settings we present the ER performance of the algorithms separately from their performance in terms of distance and popularity as ER is a measure that mainly proves that our network enrichment approach works properly, i.e., ER should increase with the increase of the influence of $\mathcal{W}_{i,j}$ on the network and the increase of the path length. Hence, based on our own measure (ER) we validate that the proposed approach works properly.

In Setting (i), for 100 randomly chosen pairs of start and target nodes the respective shortest paths within the actual road network are computed using Dijkstra's algorithm, Dij-G. Continuing, for the same start and target pairs, we run Dij-G*, Dij-H*, Dij-G** and Dij-H**. Subsequently, for each pair the difference w.r.t. ER, distance and popularity is computed, and finally averaged out over all pairs. We require the distance between start and target nodes to be at least 30% and at most 50% of the Euclidean extent of the network (approximately 6km to 10km), in order to exclude paths which start and end in the outskirts of the city (where there are few to no POIs). Figure 4.4 ((a), (c)) show the influence of the weight scaling factor $\mathcal{W}_{i,j}$ on ER for the datasets of Paris and New York respectively. As we increase $\mathcal{W}_{i,j}$, we observe an increase of ER for all four cases in comparison to Dij-G in both datasets. For the Paris dataset, the increase in ER is in the range of 80% to 250% for the Dij-G* and Dij-G**, with the latter performing better, and in the range of 250% to 620% for Dij-H* and Dij-H** with the latter performing better. For the New York dataset, the increase in ER is in the range of 20% to 80% for the Dij-G* and Dij-G** , with the latter performing better, and in the range of 80% to 150% for Dij-H* and Dij-H** , with the latter performing better.

Moreover, the first column of Figure 4.5 and Figure 4.6 ((a), (c)) shows the influence of weight scaling factor $\mathcal{W}_{i,j}$ on distance and popularity. As we increase $\mathcal{W}_{i,j}$ from 0.2 to 1.0, we observe an increase of distance and popularity for both all cases in comparison to Dij-G in both datasets. The increase among all datasets, in terms of path length is in the range of 3% to 16% for Dij-G* and Dij-G** , and in the range of 7% to 38% for Dij-H* and Dij-H**. Additionally, the increase in popularity is in the range of 30% to 120% for Dij-G* and Dij-G**, and in the range of 40% to 160% for Dij-H*and Dij-H**.

It is clear that Dij-G* and Dij-G**always perform better than Dij-H* and Dij-H** in terms of path length increase, but Dij-H* and Dij-H** perform always better in terms of ER and popularity. This is because Dij-H* and Dij-H**route directly through the POIs, causing greater detours, but passing along highly weighted parts of the enriched graphs (G^* or G^{**}), which mostly coincide with dense Flickr regions. Moreover, it is clear that *S-enrich* always performs better than *D-enrich*, in terms of ER and popularity with a very short increase, of about $2-3$% in path length. This validates that skyline enrichment provides competitive paths in terms of distance (minor increase) and popularity (significant increase).

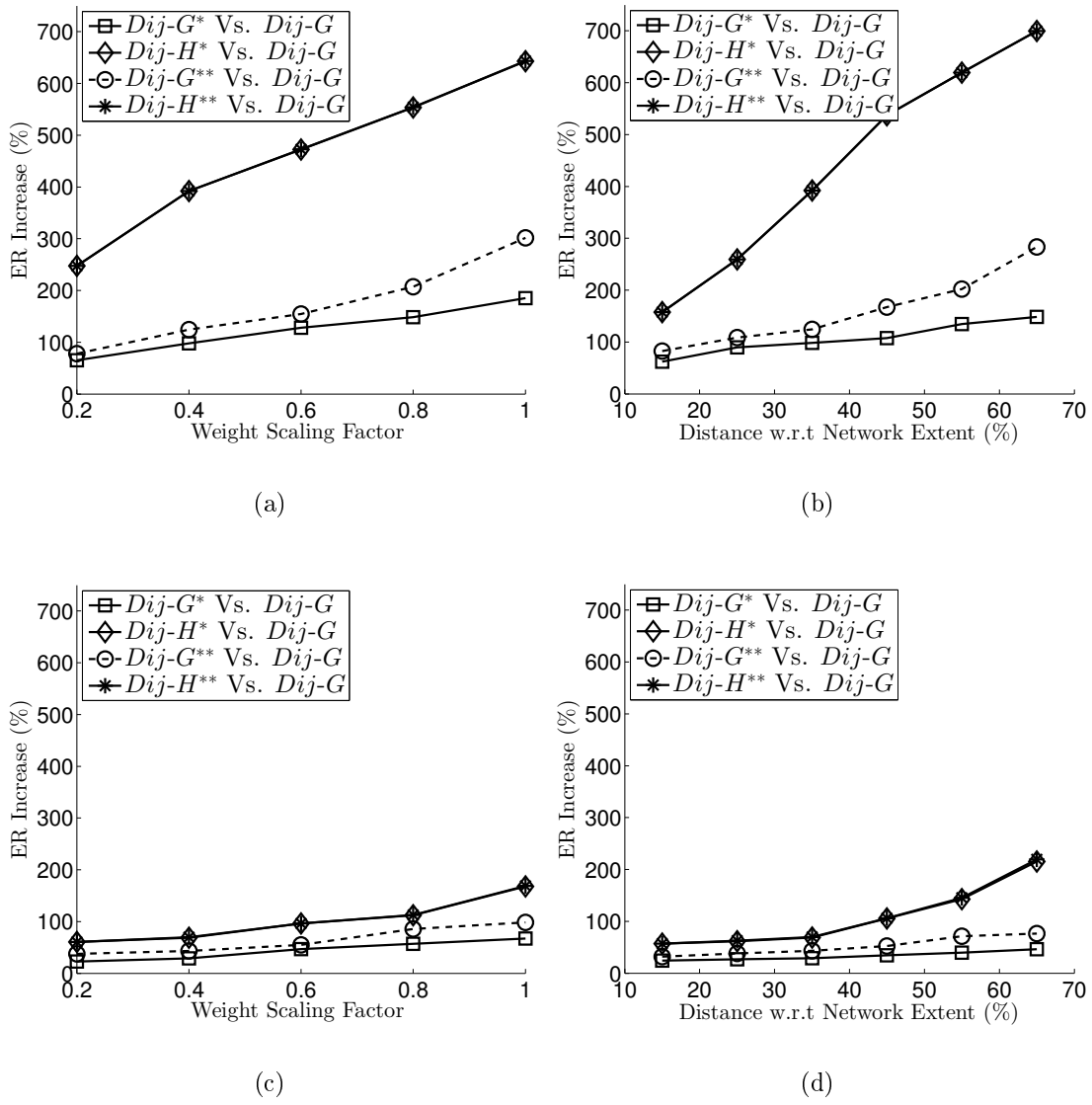


Figure 4.4: *(a), (b) show ER increase for algorithms Dij-G* and Dij-H*for Paris dataset for Settings i and ii respectively. (c), (d) show ER increase for algorithms Dij-G* and Dij-H*for New York dataset for Settings i and ii respectively.*

Continuing, in Setting (*ii*) we vary the distance of start and target relative to the extent of the whole network. We consider five different distance brackets of shortest path in the original graph G, the first one ranging from 10% to 20%, the last one ranging from 50% to 60% of the extent of the whole network. For 100 randomly chosen pairs of start and target nodes (within the respective distance bracket) paths with Dij-G , Dij-G* , Dij-G** , Dij-H* and Dij-H** are computed. As before, for each pair the difference w.r.t. ER, distance and popularity is computed and averaged out over all pairs. Figure 4.4 ((b), (d)) show the increase of ER as we proceed through the distance brackets for both datasets. The second column of Figure 4.5 and Figure 4.6 ((b), (d)) show the results in terms of distance and popularity increase. As we proceed through the distance brackets, we observe an increase of the distance and popularity for all cases in comparison to Dij-G in both datasets. The increase among all datasets, in terms of path length, is in the range

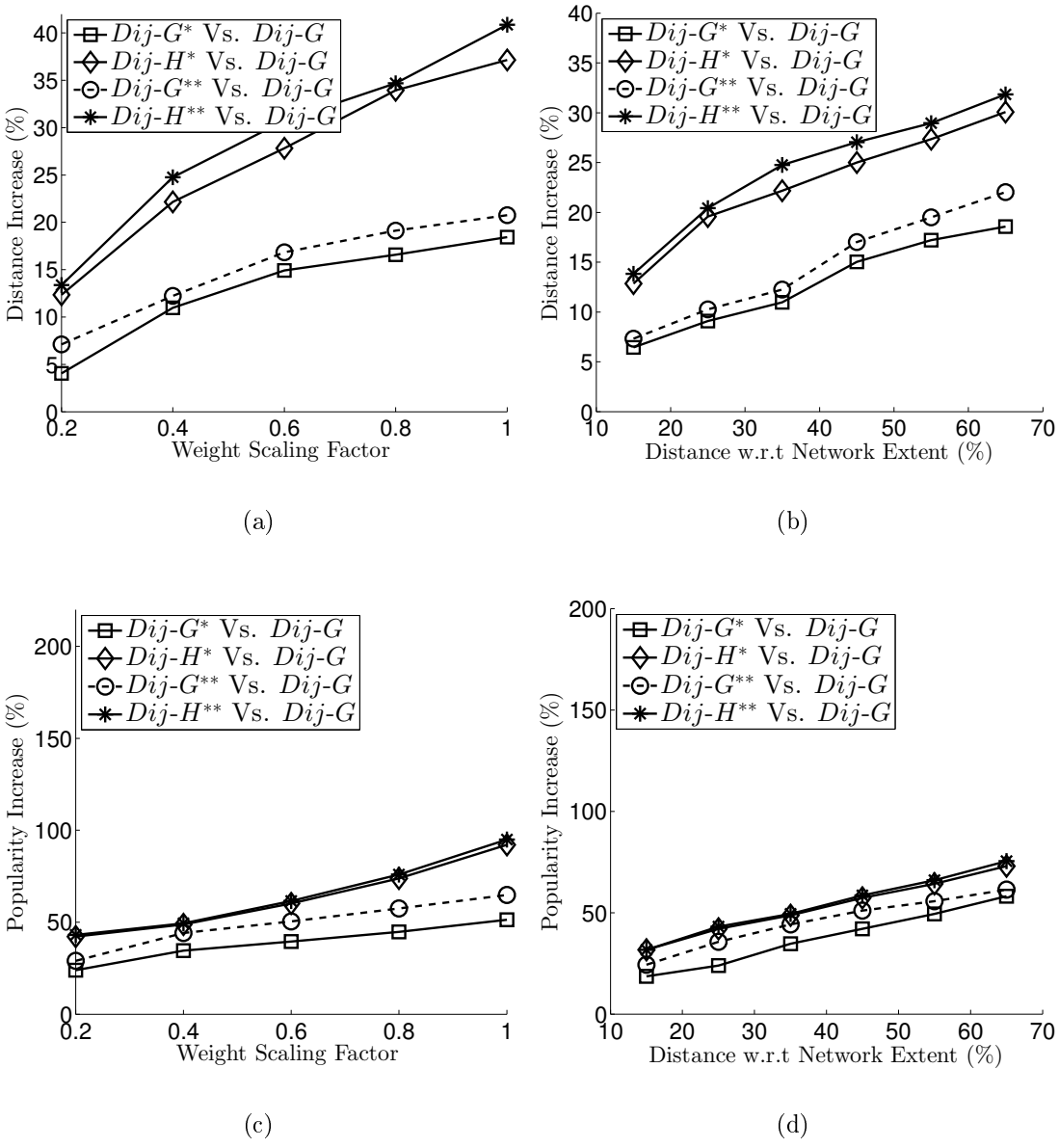


Figure 4.5: *(a), (c) show Distance and Flickr popularity increase for algorithms Dij-G* and Dij-H* for Paris dataset for experimental Setting i. (b), (d) show Distance and Flickr popularity increase for algorithms Dij-G* and Dij-H* for Paris dataset for experimental Setting ii.*

of 3% to 18% for Dij-G* and Dij-G**, and in the range of 5% to 30% for Dij-H* and Dij-H**. Finally, the increase in terms of popularity is in the range of 10% to 70% for Dij-G* and Dij-G**, and in the range of 30% to 140% for Dij-H* and Dij-H**. As in our previous experimental setting, it is clear that Dij-G* and Dij-H* always perform slightly better (only 2-3%) in terms of path length increase, while Dij-G**and Dij-H* always outperform Dij-G* and Dij-H*in terms of enrichment ratio and popularity. This underlines the validaty of *S-enrich*, as it provides significantly more popular paths while only incurring minor detours ($2-3\%$ in terms of path length).

Here, we may conclude that both *D-enrich* and *S-enrich* approaches show convincing results.Both cases yield significant increase in terms of ER as well as in terms of the independent Flickr-based measure popularity, while increasing path

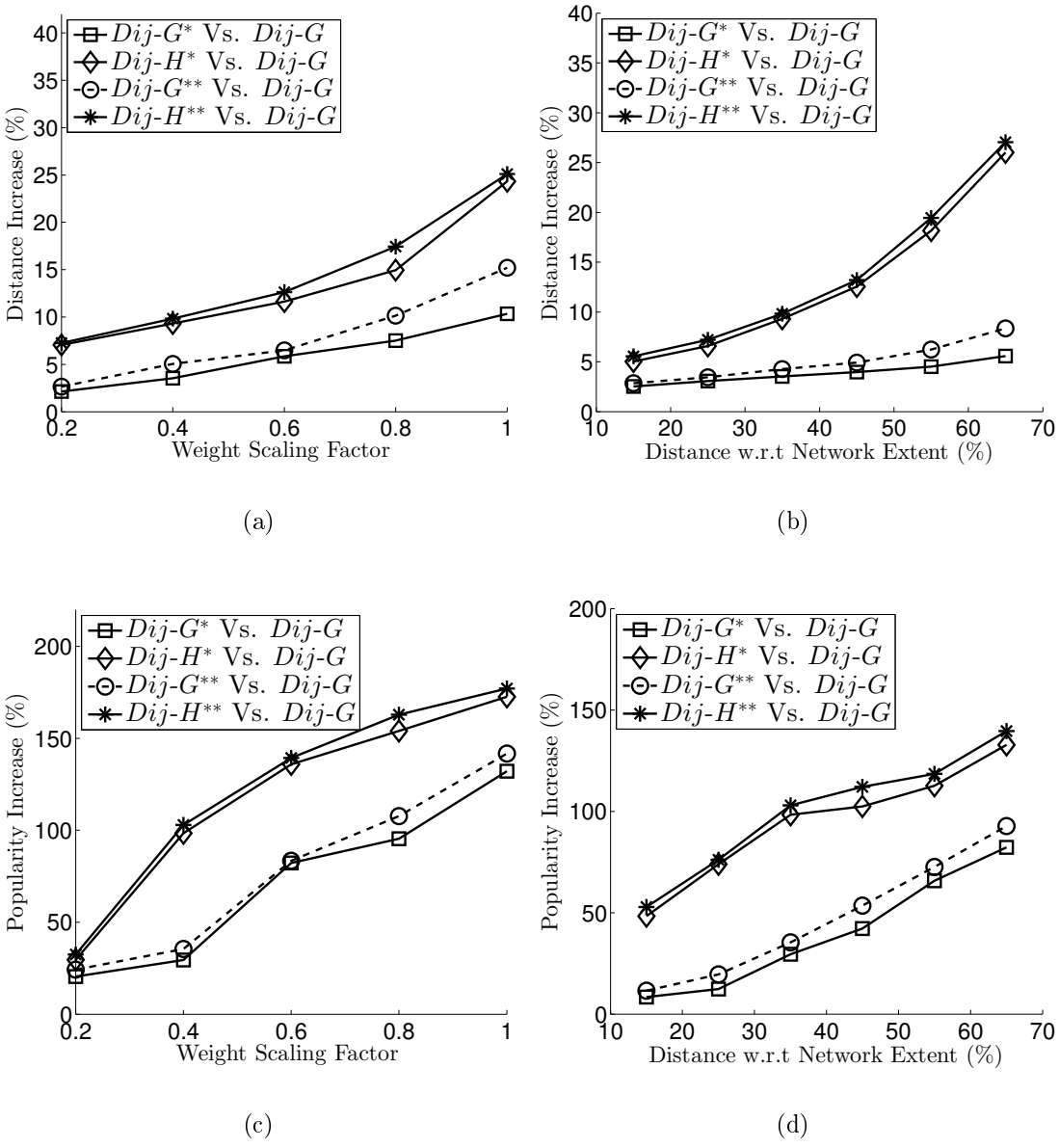
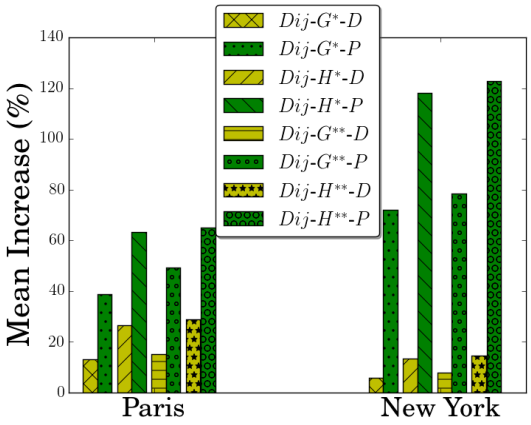


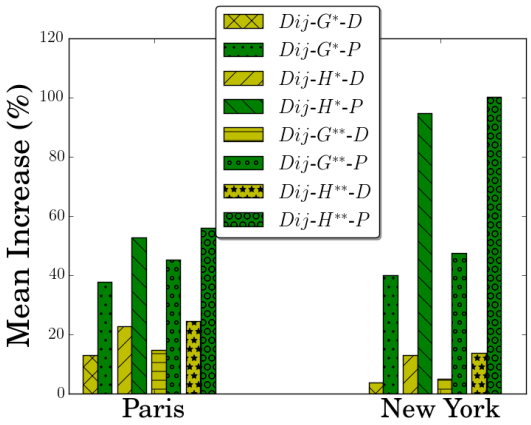
Figure 4.6: *(a), (c) show Distance and Flickr popularity increase for algorithms Dij-G* and Dij-H* for New York dataset for experimental Setting i. (b), (d) show Distance and Flickr popularity increase for algorithms Dij-G* and Dij-H* for New York dataset for experimental Setting ii.*

length only slightly. In the best case, ER increase amounts to almost 700% while popularity increase amounts to almost 160% increase (in comparison to the conventional shortest paths, as computed by Dij-G), while the worst case increase in path length is about 38% with most cases being less than 10%. Overall, *D-enrich* works slightly (2-3%) better in terms of path length while the *S-enrich* is always significantly better (more than 10% in most of the cases) in terms of popularity scores. Consequently, we can claim that spatial relations, extracted from crowd-sourced information, can indeed be used to enrich actual road networks and define an alternative kind of routing which reflects what people perceive as "close".

Finally, Figure 4.7 illustrates the trade-off (mean distance and popularity increase overall experiments) that we take by deviating from the shortest path in



(a) Setting i



(b) Setting ii

Figure 4.7: *Trade-off between distance and popularity increase of paths.*

order to obtain more interesting paths. This figure shows the relative increase in distance and popularity of the paths returned by our proposed approaches, compared to the baseline approach Dij-G. Here, we use letter D to refer to the distance increase while we use letter P to refer to popularity increase. For both datasets, we can observe that by road network enrichment we can obtain a significant increase in popularity of up to 120% for the meager price of no more than 25% additional distance incurred in both experimental settings. With the proposed *S-enrich* approach we achieve to significantly increase popularity while keeping the distance increase almost in the same levels with the *D-enrich* approach.

4.6 Demonstrations

In this part of the dissertation, we present two systems that compute popular paths based on the previously explained algorithms. Both of them use variations of the enrichment approaches presented before while in the same time they combine several kinds of user generated content during path computation. Both approaches are mainly based on skyline path computation and implemented under the Mario framework (c.f. [GKRS]).

4.6.1 A Framework for Computation of Popular Paths from Crowdsourced Data

Directions and paths, as commonly provided by route guidance systems, are usually derived considering absolute metrics, e.g., finding the shortest path within the underlying road network. This demo presents a framework which uses crowdsourced geospatial data to obtain paths that do not only minimize travel time but also guide users along popular points of interest (POIs). By analyzing textual travel blog data and Flickr data, we define a measure for popularity of POIs. This measure is used as an additional cost criterion in the underlying road network graph. Furthermore, we propose an approach to reduce the problem of finding paths which maximize popularity while minimizing travel time to the computation of bicriterion pareto optimal paths. The presented framework allows users to specify origin and destination within a road network, returning the set of pareto optimal paths or a subset thereof if a desired number of POIs along the path has been specified. Each of the returned routes is enriched with representative Flickr images and textual information from travel blogs. The framework and its results show that the computed paths yield competitive solutions in terms of travel time while also providing more "popular" paths, making routing easier and more informative for the user.

4.6.1.1 Preliminaries

User-generated content has benefited many scientific disciplines by providing a wealth of new data. The proliferation of smartphones and GPS receivers has facilitated contributing to the plethora of available information. OpenStreetMap (OSM) constitutes the standard example in the area of volunteered geographic information. Authoring geospatial information typically implies coordinate-based, *quantitative data*. Contributing quantitative data requires specialized applications (often part of social media platforms) and/or specialized knowledge, as is the case with OSM.

The majority of users contributing content, however, are much more comfortable using *qualitative information*. People usually do not use geographical coordinates to describe their favorite places or their spatial motion. Instead, it is more common for people to rely on adjectives (such as "great" or "cool") to describe their liking relative to certain points of interest (POIs). Hence, there is an abundance of largely unused geospatial information (freely) available on the internet, e.g., in travel blogs. In contrast to quantitative information, which is mathematically measurable (although sometimes flawed by measurement errors), qualitative information is based on personal cognition. Therefore, accumulated and processed qualitative information may better represent the human way of thinking and feeling.

This is of particular interest when considering the routing problem, i.e., computation of paths in road networks. Traditional routing algorithms only take the structure of the underlying road network into account, for instance, in order to compute shortest or fastest paths, i.e., optimizing w.r.t. inherently quantitative measures. However, in real life, users may be willing to find a trade-off between quantitative measures and qualitative benefit. For example:

- a tourist may be willing to take a detour in order to maximize the number and popularity of POIs on their way to the hotel,
- a commuter driving to work may prefer a slight detour if it yields a significantly nicer route,
- a dog walker might want to avoid busy and big roads altogether and favor recreational walks along landmarks and parks.

There is some existing work in this field, although most papers focus on providing paths which are easier to memorize, describe, and follow. For example, the authors of [SB13], [DK], and [WR11] try to tackle the problem by introducing cost criteria that allow for a trade-off between minimizing the length of a path while also minimizing the complexity in terms of instructions or turns along the path. The approach most similar to the one presented in this work is [QSA], which proposes a method for computing beautiful paths, as the authors phrase it. However, in order to quantify quality, the authors rely on explicit statements about the beauty of specific locations, obtained from a platform which collects user opinions on photos of specific locations. In contrast, we propose to mine this kind of information from crowdsourced data. This approach has the crucial advantage that it is scalable as the used information is already available. Having local expert users rate photos one by one, however, can hardly be extended to a global scale.

There are two general problems concerning the computation of qualitative paths. First, quality is not easily quantified. Second, the trade-off between quantitative measure and qualitative benefit is subjective and unknown to the framework. This work attempts to close this gap, by quantifying the quality of a POI by mining crowdsourced (or user-generated) data, yielding a popularity estimation. This estimation is then applied to the underlying road network as an additional cost criterion. Consequently, we obtain a multicriterion graph representing a road network (although for reasons of simplicity we focus on the bicriterion case in our demonstration). Given user input of start and target within the road network, we incorporate state-of-the-art algorithms ([SJSK], [KRS]) to efficiently compute all pareto optimal paths. Thus, the framework returns all paths between start and target which are optimal under some monotonic function, i.e., the results generated by our framework reflect all practical user preferences. These routes are furthermore enhanced by images and text. The data flow is illustrated in Figure 4.8.

The challenge of this work is to extract the crowdsourced information (from different sources) and use it to enrich an existing road network. This enriched road network is subsequently used to provide paths that satisfy the claim of high popularity (formally introduced in the next Section) while only incurring minor additional travel time.

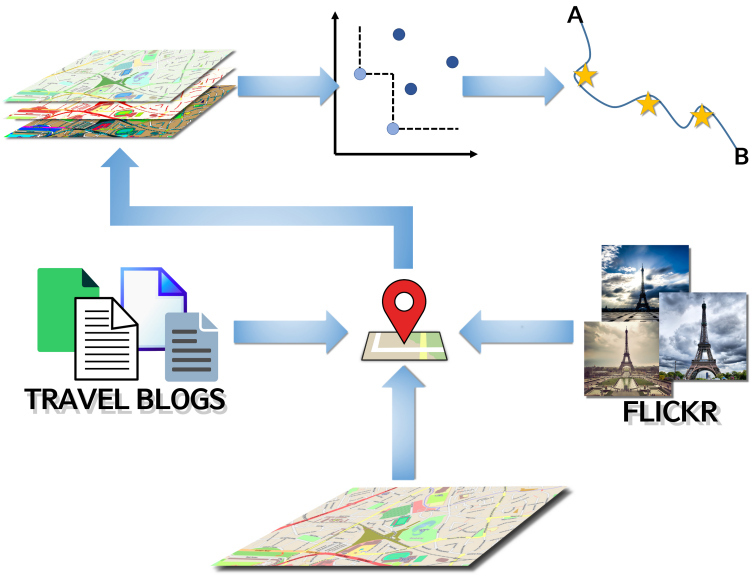


Figure 4.8: *Data flow chart of the framework, illustrating the data sources as well as the data processing.*

4.6.1.2 Contribution

In this section we describe the different steps for enriching a road network with crowdsourced geospatial information. First, we describe the sources of information and the respective extraction process. Next, we describe our approach to quantifying quality, i.e., the explicit computation of our measure for popularity estimation. Finally, we discuss our computational methods for path skylines.

4.6.1.3 Quantifying Popularity of Points of Interest

In this work, again we choose travel blogs and image datasets as a rich source for crowdsourced geospatial data. This selection is based on the fact that people tend to describe, mention, and photograph POIs that they like or find particularly interesting.

4.6.1.4 Extracting Popularity Information from Text

In a travel blog, tourists express their experiences in relation to journeys taken and places visited. Therefore, places are often associated with qualitative adjectives such as “beatiful”,“interesting”, and “cool”. To gather such data, we use classical web-crawling techniques and compile a database consisting of 250,000 texts, obtained from travel blogs as presented in [SSJ+]. Extracting qualitative information from text involves the detection of POIs which are mentioned in positive context. The employed approach involves geoparsing, i.e., the detection of candidate phrases containing references to POIs, geocoding, i.e., linking the POIs to geo-coordinates, and sentiment analysis, i.e., the evaluation of such phrases w.r.t. their connotation. Using the Natural Language Processing Toolkit (NLTK), a leading platform for analyzing raw natural language data, we managed to extract 500,000 POIs from the text corpus.

For geocoding, we rely on the GeoNames[2] geographical gazetteer database which contains over eight million POI names and their coordinates worldwide. Whenever possible, POIs extracted from the text corpus are mapped onto geo coordinates. This procedure was successful for 96% of the POIs.

Having identified and geocoded the POIs, the next step is determining the POIs mentioned in positive context. Sentiment analysis, also referred to as opinion mining, is a Natural Language Processing (NLP) problem, which has been thouroughly studied [LB02]; existing tools perform well on any kind of data. For a given POI p and every phrase mentioning p, the NLTK sentiment analysis provides a score from 0 to 1, reflecting negative to positive context. These scores are then averaged, providing a *travel blog popularity score* $\mathrm{txt}(p) \in [0,1]$ for every POI p.

4.6.1.5 Extracting Popularity Information from Images

For Flickr image data, we use the geotagged dataset provided by the authors in [MSWHD] and employ a straightforward popularity estimation approach. We assume a linear correlation between the number of Flickr images in the vicinity of a POI and its popularity within the Flickr community, and we assume this popularity to be an estimation of its general popularity. Thus, for each POI, we aggregate the number of Flickr images within an ε-range (we use $\varepsilon = 100m$ and Euclidean distance) of the POI. Let n_p denote the number of Flickr images in the ε-range of POI p, and let $N := \max n_p$ denote the maximum number Flickr images associated with any POI. Then $\mathrm{im}(p) := n_p/N \in [0,1]$ defines the *image popularity score*.

4.6.1.6 Popularity Graph Enrichment

In a next step, we want to enrich the underlying road network with a combination of both popularity scores. We investigated several strategies and decided upon a compromise between simplicity and effectiveness. Remember that we are interested in paths which provide a trade-off between a quantitative measure (in this case, travel time) and popularity scores. The main challenge is the fact that popularity is a gain, not a cost. Thus, naïve path finding algorithms, which are designed to minimize cost criteria, are not applicable. The "most popular" path would be the solution to the Traveling Salesman Problem among all POIs with a popularity gain > 0. Therefore, rather than considering vertex-associated gain, we transform the score values into edge-associated costs. We now describe this procedure in detail.

Let $G = (V, E, t)$ denote the graph representing the underlying road network, i.e., the vertices $v \in V$ correspond to crossroads, dead ends, etc., the edges $E \ni e = (u, v) \in V \times V$ represent roads connecting distinct vertices. Furthermore, let $t : E \to \mathbb{R}_0^+$ denote the function which maps every edge onto its travel time. We refer to the graph G as *road network graph*. A set of consecutive, acyclic, and mutually different edges is referred to as a *path*. Obviously, the function t naturally extends to any path r, as $t(r)$ is defined as the summed travel time of its edges. Note that our framework and its theoretical background may equally be applied to any other cost criteria (and combinations thereof). However, for reasons of simplicity, we restrict ourselves to travel time as it is probably the most essential criterion for inner city travel.

[2] http://www.geonames.org/

Let $\mathcal{P}$ denote the set of all POIs. We assume that $\mathcal{P} \subseteq V$, i.e., each POI is also a vertex in the graph. This assumption comes without loss of generality, as we can easily map each POI to the nearest node of the graph or introduce pseudo-nodes. Also, let $\hat{p}(v) := \tau \cdot \text{im}(v) + (1 - \tau) \cdot \text{txt}(v)$ denote the *popularity score* for a vertex $v \in V$. $\hat{p}(v)$ is zero if v is not a POI.

Now, we transform this vertex-associated gain into an edge-associated cost. For each edge $(u, v) = e \in E$, we define the *popularity cost* $p(e)$ as follows

$$p(e) := \phi^{(\hat{p}(u)+\hat{p}(v))/t(e)}$$

where $\phi \in]0, 1[$ is a scaling parameter. Intuitively, $p(e)$ equals 1, if e connects two vertices u and v with popularity score 0. For high popularity scores per distance of e, $p(e)$ approaches 0. Also, $p(e)$ considers the travel time of edge e, such that edges with high travel time require a higher popularity score to maintain the same $p(e)$. If a given road network graph G is enriched with a popularity cost criterion p, we refer to $G' = (V, E, t, p)$ as the *enriched (road network) graph*. Thus, in an enriched graph, every path r is assigned a two-dimensional cost vector, $(t(r), p(r))$ comprising the summed travel time and popularity costs of its edges.

4.6.1.7 Bi-Attribute Skyline Computation

As motivated in the introduction, different users may have different prioritization of the given cost criteria. In our case, depending on the type of user (e.g., tourist, dog walker, commuter), travel time is weighed against estimated popularity. Thus, we argue that without any specific knowledge of a user's preferences, we cannot guarantee any path to be optimal to the user. Therefore, we propose to return a set of alternative paths to the user, such that each alternative is pareto optimal w.r.t. to the quantitative and qualitative measures.

Given start and target nodes $s, t \in V$ in an enriched road network graph, the set of pareto optimal paths consists of all paths r between s and t which are non-dominated in the following sense: For each r there exists no other path r' (between s and t) with lower travel time and lower popularity cost, i.e., $\nexists\ r' : t(r') < t(r) \wedge p(r') < p(r)$.

This definition is, of course, a special case of the general definition of multi-criterion pareto optimality. To find all Pareto Optimal Popular Paths, we use the framwork for Multi-Attribute Routing in OpenStreetMap (MARiO) [GKRS], where the popularity cost, as defined above, is incorporated as a new cost criterion.

4.6.1.8 k-Constrained Pareto Optimal Popular Paths

A problem with pareto optimal path computation is the often extensive number of results. Especially in inner city road networks, there exist an abundance of different routing possibilities, each resulting in a slightly different cost vector. As the number of pareto optimal paths is generally unrestricted, we propose to limit the result set in order to improve usability. Also, the POIs along a path may be interpreted as path descriptors. Therefore, it may be of interest to the user to explicitly specify the number of "popular waypoints" along their desired path. Hence, we introduce the *k-constrained pareto optimal popular paths*. Choosing an appropriate k for a specific search task depends on the distance of start and target. Experimentally, a value between 3 and 6 has proven reasonable. *[k-Constrained (Pareto Optimal)*

Popular Paths] Given an integer k as well as start and target nodes $s,t \in V$ an enriched road network graph, the set of k-constrained pareto optimal popular paths consists of all popular paths that visit at least k POIs. If a path encounters more than k POIs, it is easy to determine the k most popular POIs (simply by comparing scores). Thus, a path which fits the desired parameter is obtained and returned to the user. Concludingly, by returning the k-constrained pareto optimal popular paths, we provide the user with a selection of non-dominated paths which typically encounter highly significant POIs within the respective query city.

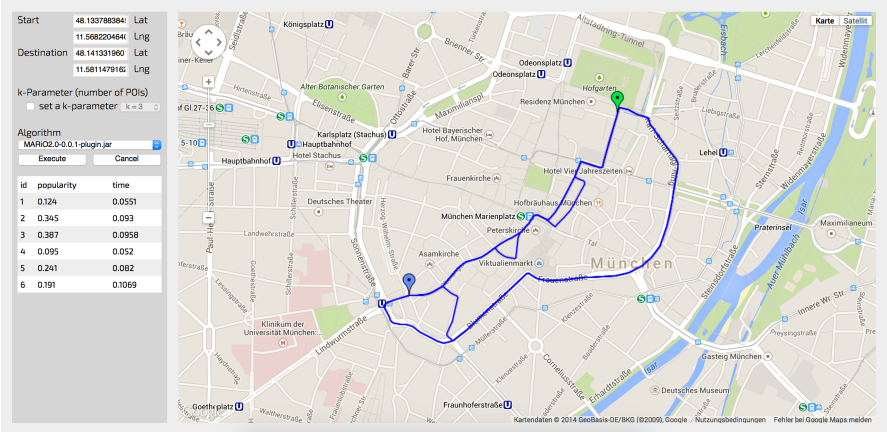
The demonstrated framework enables users to validate that the notion of popularity defined in this paper indeed coincides with the general intuition. The result paths returned to the user yield competitive solutions in terms of travel time while passing POIs perceived as significant, appealing, and/or recognizable. Hence, we solve the proclaimed task of providing "more popular" paths to the user. The two supported query types are described in Section 4.6.1.9. For both queries, the result paths are presented as a list, each with its associated costs, as well as visualized on a map relying on Google Maps. When selecting a path, the user is provided with a), selected images associated with POIs on the respective path, and b), selected travel blog entries referring to POIs on the respective path. Of course, the availibility of such images and text is dependent on the crowdsourced data and can therefore not be guaranteed. Some features of our framework are shown in Figure 4.10. Figure 4.6.2.4 depicts the main view of the framework which allows one to browse the pareto optimal bicriterion paths w.r.t. user-specified start and target nodes. The lower left corner shows a visualization of these paths, the so-called path skyline, where each path is represented by its two-dimensional cost vector. The list on the left allows to select a specific path which is then displayed on the map with a list of its respective POIs, as in Figure 4.6.2.4. As mentioned before, for each POI, additional information can be displayed. Figure 4.6.2.4 shows this information for the famous brewery Hofbräuhaus in Munich, Germany.

4.6.1.9 Supported Queries

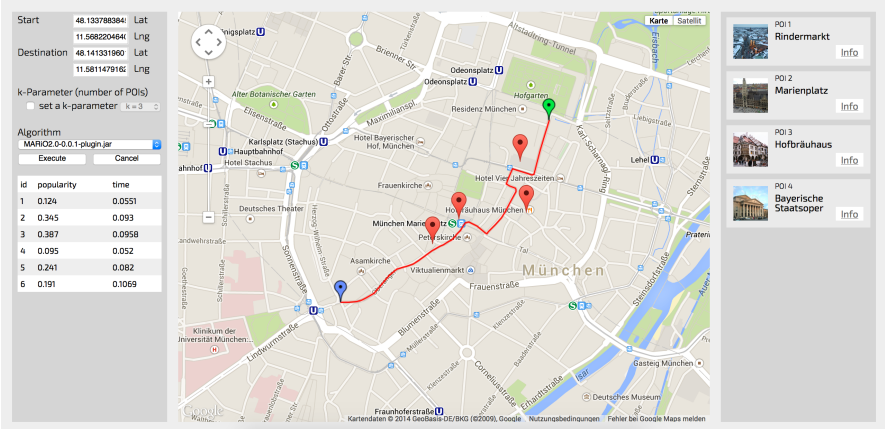
Our framework supports two different queries. The main task featured in the demonstration is the *popular path query*: Given start and target locations within a road network, our framework provides the user with the set of pareto optimal paths w.r.t. to travel time and the popularity cost. The second query allows to constrain the number of desired POIs along the path, the *k-constrained popular path query*. It returns all pareto optimal paths which visit at least k POIs and presents their k most popular POIs to the user. Obviously, the result set of a k-constrained popular path query is a subset of the result set of a popular path query with the same input. Note, however, that while the popular path query always returns at least one result, the same does not hold for the k-constrained query.

4.6.1.10 Extendability

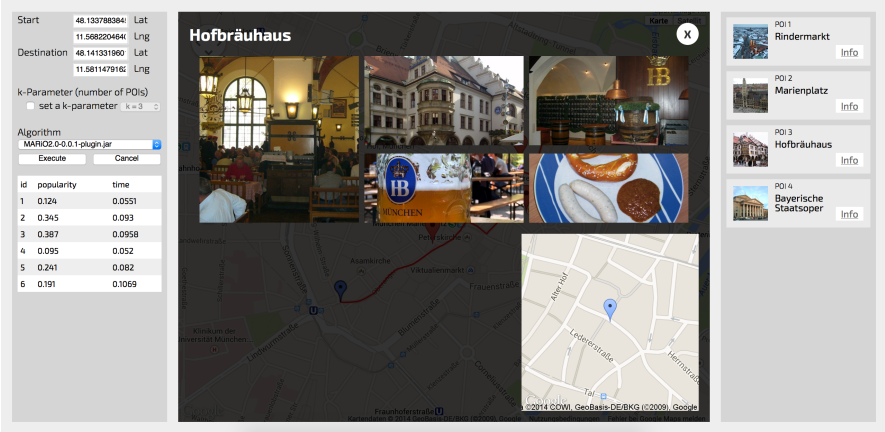
The developed framework has been built to allow easy extension and further implementation, for instance: (*i*) Other geospatial sources of crowdsourced data in order to measure the popularity of a POI, such as social check-in data, trajectory data, or other sources of textual data such as news articles can easily be incorporated. Currently, Flickr image data und textual travel blogs are supported. (*ii*) Further scoring



(a) Illustration of a Bicriterion Path Skyline



(b) Detailed Path Information



(c) Detailed POI Information

Figure 4.9: *Functionality of the presented framework.*

functions which map additional data to popularity scores can be added and the existing may be replaced. Currently, the functions presented in Section 4.6.1.4 and

Section 4.6.1.5 are used. (*iii*) The graph enrichment function which maps scores of POIs to edge-associated costs is easily interchangeable. Currently, the enrichment function of Section 4.6.1.6 is used. (*iv*) Different path skyline computation algorithms may be incorporated. Currently, the framework relies on algorithms implemented in the the MARiO framework [GKRS].

4.6.2 Tourismo: User Preference Driven Touristic (Trip) Search Engine

In this demonstration we re-visit the problem of finding an optimal route from location A to B. Currently, navigation systems compute shortest, fastest, most economic routes or any combination thereof. More often than not users want to consider "soft" qualitative metrics such as popularity, scenic value, and general appeal of a route. Routing algorithms have not (yet) been able to appreciate, measure, and evaluate such qualitative measures. Given the emergence of user-generated content, data exists that records user preference. This work exploits user-generated data, including image data, text data and trajectory data, to estimate the attractiveness of parts of the spatial network in relation to a particular user. We enrich the spatial network dataset by quantitative scores reflecting qualitative attractiveness. These scores are derived from a user-specific self-assessment ("On vacation I am interested in: family entertainment, cultural activities, exotic food") and the selection of a respective subset of existing POIs. Using the enriched network, our demonstrator allows to perform a bicriterion optimal path search, which optimizes both travel time as well as the attractiveness of the route. Users will be able to choose from a whole skyline of alternative routes based on their preference. A chosen route will also be illustrated using user-generated data, such as images, textual narrative, and trajectories, i.e., data that showcase attractiveness and hopefully lead to a perfect trip.

4.6.2.1 Preliminaries

Nowadays, social networks are a great source of rich geo-spatial data. Almost every social network allows users to incorporate geo-social features into their data stream. The different features include, amongst others, geo-tagged pictures (e.g. Flickr), geo-descriptive text (e.g. travel blogs), and tracked movement (e.g., runners' trajectories). For this demo, we rely on all these kinds of user-generated data to define attractiveness on a real world road network. Our aim is to reflect human fondness according to the crowd by using qualitative information and making it measurable. We present *Tourismo*, a tourist search engine, which computes attractive paths along points of interest (POIs), tailored to the interest of the user issuing the query. Based on this enriched spatial network, which has information about the attractiveness of locations, we aim at answering *attractive path queries*. Currently, navigation systems, i.e., machines, perform this task for us, computing routes such as the shortest route, the fastest route, the most economic route [AJTY13], or some combination of such quantitative measure on a spatial network [GKRS]. In all of these cases, the employed algorithms optimize cost measures inherent in the underlying road network. What is rarely reflected, however, is user preference on subjective measures, such as attractiveness and interestingness of a route. Often users are willing to take a suboptimal detour, a deviation from quantitative optimality (shortest, fastest, etc.),

in order to improve the quality of their route. In order to see more attractions, for instance, a tourist may be willing to take a moderate detour from a fast, but not very attractive, highway.

How can we measure a subjective concept of "quality"? How to measure attractive, scenic, recreative routes? As machines are not (yet) capable to reflect this concept, we rely on the crowd to answer this question, i.e., we propose to use crowdsourced data to estimate the attractiveness of an area. Relying on different datasets, image data (from Flickr[3]), textual narratives (from travel blogs), and trajectory data (from Endomondo [4]), we investigate the applicability of different data sources as cost measures for the underlying road network. More precisely, we enrich the road network by quantitative scores of qualitative statements as follows:

- areas having a large density of Flickr images indicate a particularly attractive area, increasing the attractiveness score;
- locations mentioned in the positive context of travel blogs increase attractiveness scores;
- routes commonly used by other users are also considered more attractive.

Furthermore, we incorporate meta-information from OpenStreetMap[5], in order to categorize POIs and, using the aforementioned popoularity score, propose routes according to the user's preferences and the fondness of the crowd. *Tourismo* presents solutions to enrich the underlying road network using the aforementioned data sources. We show an initial approach to map these *attractiveness scores* to a cost measure, which allows one to apply existing routing algorithms which aim at minimizing edge-labeled cost metrics. We apply an adapted algorithm for bicriterion pareto-optimal route search, to find paths which are optimal in both travel time and attractiveness. Our framework allows to specify origin and destination, computes and displays the skyline of pareto-optimal paths. Furthermore, the reasons for attractiveness of each path are illustrated: Flickr images along the way, travel blog entries mentioning locations on the way, and historical trajectories which share the same route. Our demonstrator, which we would like to present to the community at SSTD'15, is an extension of a demonstrator that we recently presented at ICDE'15 [JFS+15].[6] The new demonstrator has two major features: First, our demonstration allows to specify the interest of a user, thus allowing to return routes that contain POIs which are of particular interest to the user issuing the query. Second, this version allows to consider a third type of data to enrich the underlying road network with attractiveness information: In addition to geotagged images, and texts containing geospatial references, we allow to learn attractiveness from an existing base of historic trajectory data.

4.6.2.2 State of the Art

Recently, a lot of interesting research has been done in the context of finding scenic, interesting or popular routes. The first set of related work focuses on providing

[3] www.flickr.com

[4] www.endomondo.com

[5] www.openstreetmap.org

[6] Since the ICDE proceedings are not publicly accessible at this time, we have attached the demonstration proposal at the end of this submission. Clearly, this footnote and the attached paper will be removed for a potential camera ready.

paths which are easier to memorize, describe, and follow. For example, the authors of [SB13], [DK], and [WR11] try to tackle the problem by introducing cost criteria that allow for a trade-off between minimizing the length of a path while also minimizing the complexity in terms of instructions or turns along the path. Furthermore, an existing research direction covers the problem of defining tourist routes, which maximize the subset of a set of pre-defined POIs which can be visited in a tourist tour that has a time-constraint [GAL+10, GKMP14]. In these works, the set of interesting POIs is given, and the main conceptual contribution of is to automatically extract interesting locations, as well as a quantitative estimate of the popularity of this location from a variety of data sources.

The approach most similar to the one presented in this work is [QSA], which proposes a method for computing beautiful paths, as the authors phrase it. However, in order to quantify quality, the authors rely on explicit statements about the beauty of specific locations, obtained from a crowd-sourcing platform which collects user opinions on photos of specific locations. In contrast, we propose to mine this kind of information from existing crowd-sourced data, which does not require any monetary investment to aquire. Thus our approach has the crucial advantage that it is scalable as the used data is already available globally available, while having local expert users rate photos one by one can hardly be extended to a global scale.

Another important research direction is the *stitching* existing trajectories in order to obtain new trajectories which guarantee that each sub-trajectory is used by other users, and is thus, "popular" following the definition [CSZ11] of Chen et al. This, however, only reflects a notion common usage, not taking into account, why a specific sub-trajectory has been favored. For instance, when mining trajectories of commuters, the fastest path is most likely to be chosen by most users. Hence, we propose mining trajectories specific to recreational use and merging this information with the attractiveness scores we derive from other user-generated data sources.

4.6.2.3 Features

The main feature of this demonstrator is the estimation of attractiveness from text, image, and trajectory data. Details covering text and image data can be found in [JFS+15]. In this section, we briefly describe how we enrich the underlying road network using historical trajectory data. For our demonstrator, we use trajectories of walkers, runners and bikers that have uploaded their workouts to Endomondo. Our dataset contains eight million trajectories, which are located all around the world, but have a strong regional focus in Northern Europe. To match each of the GPS trajectories, we apply state-of-the-art map matching techniques, similar to those presented in [NK09]. In a first step, we perform a basic enrichment: For each edge e of the spatial network, we count the number $\mathrm{tra}(e)$ of historical trajectories that contain this edge. This count can be used as an indication of attractiveness of an edge, following the assumption that runners are, in average, more likely to choose a particularly nice running trail. As mentioned before, following the techniques proposed in [JFS+15], we obtain a road network having a attractiveness score derived from Flickr image data and travel blog text data. On top of that, we add the trajectory attractiveness $\mathrm{tra}(e)$, by introducing a new user-specific weighting factor which is omitted here for reasons of brevity. Relying on the enriched road network, *Tourismo* supports bicriterion pareto-optimal path queries.

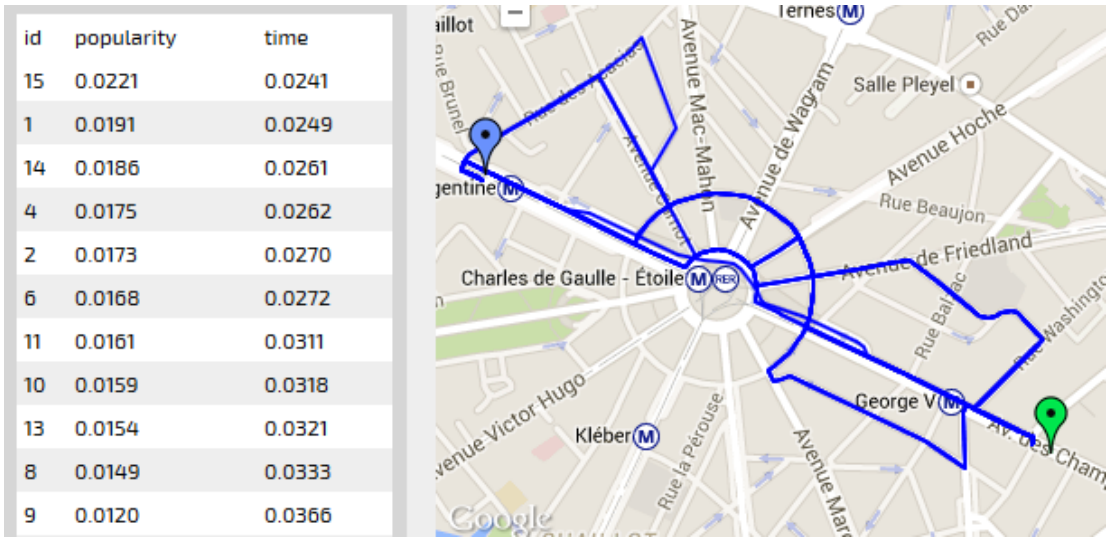
Additionally, *Tourismo* features category-specific path queries. If the user chooses

to specify his personal touristic interests, they can choose one or more options from a list containing outdoor activities, cultural sightseeing, culinary interest, and more. In order to provide paths which fulfill these requirements, we mine the meta-information provided by OSM. Thanks to a very active community, OSM data contains well-tended information about POIs, that is named, categorized, and subcategorized. For instance, the meta-categories "food" and "tourist" contain subcategories "bar", "restaurant", "fastfood" and "monument", "museum", "archeological", respectively. Mapping these categories onto the options of user-preferences, we are able to filter POIs which correlate to the particular interest of the user. When querying a route with a specific set of interests, the user is provided a number of pareto-optimal paths, guiding him along POIs tailored to his preference.

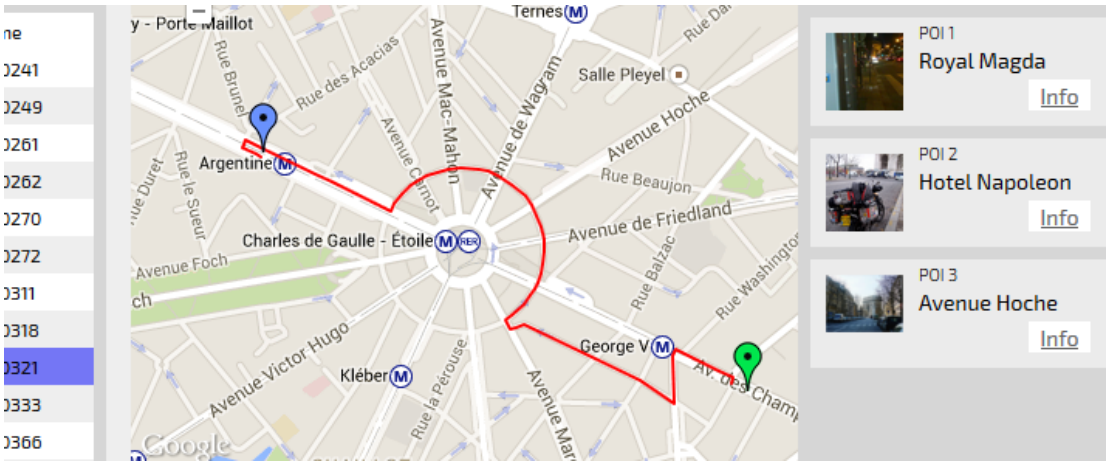
4.6.2.4 Framework Description

The demonstrated framework allows users to validate that the notion of attractiveness defined in this paper indeed coincides with the general intuition. The result paths returned to the user yield competitive solutions in terms of travel time while passing POIs perceived as significant, appealing, and/or recognizable. Hence, we solve the proclaimed task of providing "more attractive" paths to the user. Using OpenStreetMap as a road network, our demonstrator visualizes a map relying on Google Maps. Upon selecting an origin and a destination location on the map, the user is presented with the skyline view as shown in Figure 4.10 (a). In this view, the route skyline is presented to the user, i.e., the set of routes which are pareto-optimal in terms of both popularity and travel time. For each such route, the corresponding travel times are shown in a table in the lower left corner of Figure 4.10 (a). These routes are sorted by both popularity and travel time, which is equivalent by definition of pareto-optimality, i.e., there exists no two routes A, B in the route skyline such that A is both faster and more popular than B. Using this table, the user can select a route which corresponds to the user's preference between travel time and popularity, yielding the route view shown in Figure 4.10 (b). For the selected route A, this view shows the most popular points of interest on A.

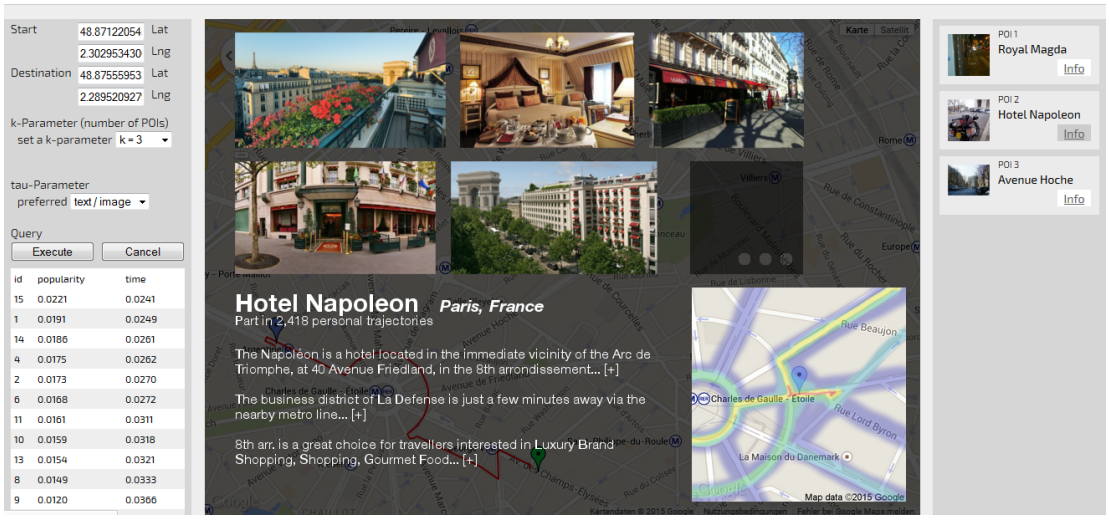
Once a point of interest is selected, the sources of popularity of this POI are shown as in Figure 4.10 (c). For this purpose, Figure 4.10 (c) shows all the pictures relevant for the selected POI, i.e., the set of images having a sufficiently low distance. The bottom-left corner shows all travel blog entries where this entry was mentioned in a positive context. Finally, the lower left corner shows a heatmap derived from all trajectories that share the same trajectory. During the demonstration, users will be able to specify start and target locations (and, if desired, specific categories of interest) on the presented web interface, e.g., their home and their office. Upon being presented with the popular path skyline, the users may browse different paths and inspect the POIs as well as the additional crowd-sourced information, including the images of PoIs on the route, travel blog entries mentioning POIs on the route, and a heat-map of trajectories covering the route.



(a) Bicriterion Path Skyline



(b) Detailed Path Information



(c) Detailed information about selected PoI

Figure 4.10: *Functionality of the presented framework.*

Chapter 5

Mining GPS Data

5.1 Non-Continuous Monitoring of KNN Trajectories

Nowadays, massive amounts of tracking data for various types of moving objects, including vehicles, humans and animals, are becoming available. Analyzing this type of spatio-temporal data is crucial for discovering movement patterns, understanding and forecasting behaviors, and developing novel applications and services. One problem of particular interest is finding objects that move close together with a certain object during some periods of time. In this part of the disseration, we focus on finding the k-nearest moving neighbors for a given query object and time interval. We formulate the problem, using a similarity function that takes into consideration both the proximity and the direction of the trajectories, and we firstly present an exact algorithm. Then, we focus on approximate algorithms in order to reduce the execution time, investigating two directions. The first employs line simplification to approximate the compared trajectories, thus reducing the calculations needed to identify the nearest neighbors. The second relies on estimates of prior probabilities derived from trajectory distributions and attempts to achieve a faster approximation of the k-nearest neighbors. A detailed experimental evaluation of the aforementioned algorithms on three real-world datasets is finally presented in order to verify their efficiency and accuracy.

5.1.1 Preliminaries

During the last years there has been a notable increase in the popularity and pervasiveness of positioning technologies (GPS-equipped devices, GSM localization, Wi-Fi, Bluetooth, RFID, etc.). This has made possible the collection of massive amounts of tracking data for various types of moving objects, including vehicles, humans and animals. Analyzing this kind of spatio-temporal data enables the discovery of useful movement patterns, which makes possible to understand and forecast behaviors and to develop novel applications and services.

To this direction, a lot of efforts have focused on mining moving object trajectories [GLW08]. Of particular interest is the task of finding objects that move close together in both space and time. This is usually addressed by searching for clusters of objects that move together for sufficiently long periods of time. Different variations, such as flocks [GvK06b, VBT09a], convoys [JYZ+08a, JSZ08], moving

clusters [KMB05a, WLH06] and swarms [LDHK10a], have been studied. Their differences lie on how the clusters are defined and on whether the objects stay together for consecutive time periods. Other works have focused on mining hidden periodic patterns in the movement behavior of objects [CMC07, LDH+10], detecting interactions (e.g., an object following another) [GLW08, AGLW08], finding outliers [LHL08] or making predictions [MPTG09].

In this part of the disseration, we are interested in finding the nearest objects that move close together with a given query object, within a query time window. The query object may be an actual moving object among those in the dataset or a virtual object describing a desired movement. The former case is useful, for example, for characterizing and classifying the behavior of the query object based on the type of its identified closest neighbors or for using these neighbors to make predictions and recommendations. In the latter case, one could specify a movement of interest and then search for those objects that most closely resemble it. Potential applications include recommending sites and events to travelers with similar behavior, tracing the spread of an infectious disease in flocks of animals, etc.

Existing approaches in the literature for k-nearest neighbor or similar types of queries on moving objects, typically focus on the continuous case (e.g., [GBX10a, GLC+07b, GLC+07a]). These queries maintain, for each time point in the query interval, the set of k-nearest neighbors to the query object, which may be either static or also moving. However, this level of granularity is often too fine-grained. Indeed, if the k-nearest neighbors of the query object change very frequently, this results in a high number of objects that need to be monitored. Hence, a subsequent mining step is then needed to identify the truly interesting ones among them. Here, we focus on the non-continuous case. The query is executed once for the given time interval and it returns an overall set of k-nearest neighbors for the whole duration specified.

The movement of an object is tracked by recording its location at various points in time. The trajectory of the object is then constructed by means of (typically linear) interpolation between the recorded positions, resulting in a series of line segments. Comparing two objects is done by comparing their traversed line segments along the time dimension. Considering the whole segments, instead of simply their endpoints, captures the objects' movement more accurately as it takes direction also into account.

Performing an exhaustive search to find the k-nearest neighbors of the query object in the requested time window is not efficient. In this part of the disseration, we firstly describe an optimized exact algorithm that speeds up the search by maintaining a priority queue. This restricts the examination to the most promising candidates, allowing the search to terminate earlier. Then, we turn to approximate algorithms, investigating two directions. The first relies on a line simplification algorithm to approximate each trajectory with just a few line segments, reducing the number of calculations. The second pre-computes probability density functions of trajectory line segments and utilizes them to estimate, at query time, prior probabilities of moving objects being close to the query point.

The main contributions of this part of the disseration are summarized below.

- We formulate the problem of finding the k-nearest moving neighbors of a moving query point within a given time interval.

- We present an exact algorithm using a priority queue to compute the k-NNs more efficiently and progressively.
- We combine the exact algorithm with a line simplification process to allow for faster computation of an approximate set of k-NNs.
- We derive a second approximate algorithm by incorporating the estimation of prior probabilities of trajectory line segments being close to the respective query line segments. This provides an alternative method for accelerating the identification of the k-NNs.
- We conduct a detailed experimental evaluation of the proposed algorithms to test both their efficiency and accuracy on real-world trajectory datasets.

5.1.2 Related Work

In the following, we review related work on trajectory similarity, moving object clusters, and k-nearest neighbor queries on moving objects.

5.1.2.1 Trajectory Similarity

Many approaches for defining similarity measures for trajectories are derived from corresponding techniques in time series analysis or from the concept of edit distance. Dynamic Time Warping (DTW) [BC96] is a method for finding the optimal match between two sequences with potentially different length. A similarity function based on the notion of the Longest Common Subsequence (LCSS), a variation of the edit distance, is proposed in [VGK02b]. The basic idea is to match two sequences by allowing them to stretch, without rearranging the sequence of the elements but allowing some elements to be unmatched. The Edit Distance on Real sequence (EDR) [CÖO05a] measures the similarity between two trajectories also based on edit distance on strings. It quantizes the distance between a pair of elements to two values to reduce noise effects, and it then seeks the minimum number of edit operations required to change one trajectory to another. These approaches aim at providing a higher degree of flexibility, e.g. to cope with varying length and speed or unmatched parts, which however is not the goal in our case.

Most typical approaches model trajectories as sequences of line segments, ignoring the time dimension, and compare them based on their shape. Such shape-based similarity queries for trajectory databases have been studied in [YAS03a]. Trajectories are modeled as directed lines in space, and their similarity is defined as the Euclidean distance between directed discrete lines. One Way Distance [LS08] also compares the spatial shapes of trajectories without taking time information into account. A trajectory is represented as a sequence of line segments, and the distance from a trajectory T_1 to a trajectory T_2 is defined as the integral of the distance from points of T_1 to T_2 divided by the length of T_1. Similarity measures for trajectories that take also the time into account have been proposed in [FGT07] and [PKM+07b].

Since it is not our goal in this work to propose a new similarity measure for comparing trajectories, we have adopted the approach used in [LHW07a]. Trajectories are represented as sequences of line segments. The distance function between two

line segments comprises three components: the perpendicular distance, the parallel distance and the angle distance, thus taking both proximity and direction into account. The time parameter is considered by comparing line segments between matching time intervals.

5.1.2.2 Moving Object Groups

Several works have studied the problem of finding groups of objects that move together. Various definitions have been proposed for capturing variants of this notion, introducing also efficient mining algorithms to detect them.

The concept of flocks of moving objects is formalized and investigated in [GvK06b, VBT09a]. Consider a set of trajectories of objects in the plane, where each trajectory consists of a sequence of line segments. A flock is defined as a group containing at least m objects, for which there exists a time interval comprising at least k consecutive time points, so that for each point in this interval there exists a disk of radius at most r containing all the m objects. Convoys [JYZ+08a, JSZ08] are defined similarly to flocks. The difference is that the objects belong to density-connected clusters, w.r.t. to a distance threshold e, instead of being contained in a disk of radius r. This allows for arbitrary shapes while still enforcing proximity.

Some other variants employ more flexible definitions. A moving object cluster [KMB05a] is a sequence of spatial clusters appearing during consecutive time points, such that the portion of common objects in any two consecutive clusters is above a given threshold θ. This allows for some members of the cluster to change during time. Group patterns [WLH06] and swarms [LDHK10a] detect objects that move close together for sufficiently large periods of time but not necessarily within a single, consecutive time interval. This permits some objects to leave the group temporarily.

Despite the similarities, these works detect groups of objects that move together, while we are interested in finding the k-nearest neighbors of a given moving object.

5.1.2.3 k-NN Queries on Moving Objects

Different types of k-NN queries over historical trajectories of moving objects are studied in [FGPT07a]. Queries are distinguished according to two factors: (a) whether the query object is stationary or moving and (b) whether the query is evaluated on a non-continuous or continuous fashion, thus resulting in four different types of nearest neighbor queries. In the non-continuous case, a single result set is computed for the whole query interval, considering the lowest distance between objects during it. In the continuous case, the result set is updated for each point in time in the specified interval. For example, for the 1-NN case, the stationary, non-continuous query returns the object that came closer to a given point Q within a time interval T, while the moving, continuous query reports, for each time snapshot within T, the object which is closer to the current location of the moving query object.

The continuous k-NN query in [GBX10a] returns all the objects which, at some point in time within the specified query time interval, belong to the k closest objects to the query object. The trajectories are indexed using a 3D-R-tree and a filter-and-refine strategy is employed. R-tree-like structures are also used in [GLC+07b, GLC+07a] for processing historical continuous k-nearest neighbor queries. Other

works have focused on processing k-NN queries on moving objects with uncertainty [HLL09, TTC+11], as well as on reverse nearest neighbor queries [LC09, CZLZ12].

In this work, we are interested in the non-continuous case, but based on a total score computed over the whole query interval. Moreover, we are interested in both exact and approximate solutions.

5.1.3 Problem Definition

Let D be a database containing N moving object trajectories. Similar to other approaches in the literature (e.g. [PKM+07b]), linear interpolation between sampled locations is assumed and the trajectory T_i of a moving object O_i is represented as a sequence of 3D line segments, where each endpoint p_j is a triple (t_j, x_j, y_j) denoting the location of object O_i at time t_j.

The spatio-temporal similarity of two trajectories is derived by aggregating the distances of their respective line segments between the corresponding timestamps. Assume a line segment L_i, with endpoints p_i, p_{i+1}, and a line segment L_j, with endpoints p_j, p_{j+1}. Also, let p'_j and p'_{j+1} denote the projection of p_j and p_{j+1} on L_i, respectively, and θ the angle between L_i and L_j. We use the distance function defined in [LHW07a], which is composed of three components:

- the *perpendicular distance*, defined as:

$$d_\perp = \frac{||p_j - p'_j||^2 + ||p_{j+1} - p'_{j+1}||^2}{||p_j - p'_j|| + ||p_{j+1} - p'_{j+1}||} \tag{5.1}$$

- the *parallel distance*, defined as:

$$d_\parallel = \min(||p_i - p'_j||, ||p_{i+1} - p'_{j+1}||) \tag{5.2}$$

- and the *angle distance*, defined as:

$$d_\theta = ||L_j|| \cdot \sin(\theta) \tag{5.3}$$

The distance between L_i and L_j is then computed as the weighted sum of the above three terms:

$$Dist(L_i, L_j) = w_\perp \cdot d_\perp + w_\parallel \cdot d_\parallel + w_\theta \cdot d_\theta \tag{5.4}$$

The advantage of using this distance function instead of simply computing the Euclidean distance of the endpoints of the segments is that it takes into account not only proximity but also differences in length and orientation. In our experiments, we have weighted these factors equally.

Furthermore, to obtain a normalized similarity value in $[0, 1]$, we define the similarity between two line segments L_i and L_j as:

$$Sim(L_i, L_j) = \frac{1}{1 + Dist(L_i, L_j)} \tag{5.5}$$

Given a time interval $[t_s, t_e]$ and two trajectories T_i and T_j defined in this interval, we can now define their similarity as the average similarity of their respective line segments, i.e.

$$Sim(T_i, T_j, s, e) = \sum_{c=s}^{e-1} \frac{Sim(L_i^c, L_j^c)}{e - s} \tag{5.6}$$

where L_i^c is the line segment of trajectory T_i between timestamps t_c and t_{c+1}.

Problem Statement. Assume a set of trajectories D, a query trajectory T_q and a query time interval $[t_s, t_e]$. The k-nearest moving neighbors of T_q during $[t_s, t_e]$ is a subset D_k of D with size k, such that: $\forall T_i \in D_k, \forall T_j \in D \setminus D_k : Sim(T_i, T_q, s, e) \geq Sim(T_j, T_q, s, e)$.

5.1.4 Computing Nearest Moving Neighbors

In this section we introduce our algorithms for solving the k-nearest moving neighbors problem defined above. We start by describing an exact k-NN algorithm, which computes the distances between the line segments of the query trajectory and those of other trajectories in the database, maintaining a priority queue to progressively identify and report the k-NNs. Then, we devise two approximate algorithms that significantly reduce the execution time while still approximating the set of k-NNs with sufficient accuracy.

5.1.5 Exact Algorithm

Finding the k-nearest moving neighbors of a moving query object in a given time window can be considered as a rank aggregation problem [FLN03]. For each pair of consecutive timestamps in the query time interval, the moving objects are sorted according to the distance of their corresponding line segment to the query line segment; then, the lists are aggregated to produce the overall k-nearest neighbors. However, this is inefficient, as it compares the whole trajectories of all objects to the query trajectory. Instead, our goal is to identify some promising candidates and focus the search on them so that it can terminate earlier.

For this purpose, we use the algorithm outlined in Algorithm 5. The idea is to maintain an upper and lower bound of the similarity of each object to the query. Every time the actual similarity of a line segment is computed, these bounds are updated accordingly, until the k objects with the highest similarity can be safely determined.

More specifically, the algorithm proceeds as follows. First, it computes the similarity of the objects to the query for the first pair of timestamps in the query interval (line 7). Based on this, it initializes the lower and upper bounds of the similarity (lines 5–8) and it organizes the objects in a priority queue according to the upper bound of their similarity (line 10). For each object, it also maintains a counter denoting the last timestamp that has been examined (line 9). Then, it retrieves the first object from the queue (line 13). It finds its line segment corresponding to the next timestamp that has not been examined, and it computes the similarity to its respective line segment of the query trajectory (line 16). Based on that, it updates the similarity bounds of this object to the query (line 17), and it reinserts it to the

priority queue, updating also the counter (lines 18–19). Once an object is retrieved from the queue for which the lower bound of the similarity is greater than or equal to the upper bound of the next object in the queue, this object is added to the output (line 14). The process is repeated until k objects have been returned.

ALGORITHM 5: Exact computation of k-nearest moving neighbors

Input: A database of trajectories D, a query trajectory Q, a query time interval $[t_s, t_e]$

Output: The k-nearest moving neighbors D_k

```
1  begin
2      D_k ← ∅
3      H ← ∅                          /* priority queue */
4      foreach P ∈ D do
5          sim_p^- ← 0
6          sim_p^+ ← 1
7          r ← Sim(L_q^s, L_p^s)
8          UpdateBounds(P, r)
9          t_p ← s + 1
10         H ← insert P in descending order of sim_p^+
11     end
12     while |D_k| < k do
13         P ← H[0]
14         if sim_p^- ≥ sim_H[1]^+ then D_k ← add P
15         else
16             r ← Sim(L_q^{t_p}, L_p^{t_p})
17             UpdateBounds(P, r)
18             t_p ← t_p + 1
19             H ← insert P in descending order of sim_p^+
20         end
21     end
22     return D_k
23 end
24 Function UpdateBounds(P, r)
25     begin
26         sim_p^- ← sim_p^- + r/(e−s)
27         sim_p^+ ← sim_p^+ − (1−r)/(e−s)
28     end
```

Algorithm 5 correctly identifies the k-nearest moving neighbors; moreover, it does so progressively[1]. Indeed, each time an object is popped from the queue, examined

[1]If it is not required to return the k-NNs in the correct order, it is possible to further speedup the process by replacing $H[1]$ with $H[k - |D_k|]$ in line 14.

and re-inserted, the bounds of its similarity are refined. In the worst case, all its line segments will be examined, and then both bounds will converge to the exact value of the similarity. Given that the objects are sorted in descending order of the upper bound of their similarity, if the lower bound of the top object is higher than the upper bound of the next, it can be safely returned.

Notice that a spatio-temporal index, such as an STR-tree or a TB-tree [PJT00a], can be used to efficiently retrieve from the database those (sub-)trajectories with line segments contained in the time interval specified by the query. Once this set of candidates is loaded, the algorithm described above is executed to compute the k-nearest neighbors. The reason that we cannot rely solely on the index to perform a k-NN search is twofold. First, the distance function between line segments is a composite one, involving both Euclidean distances between points and angle distances, hence the standard MBR-based pruning is not suitable. Second, the set of k-nearest neighbors is not monotonic, i.e., an object that belongs to the k-NN set for a time interval $[t_i, t_j]$ may not belong to the k-NN set of any time interval $[t_{i'}, t_{j'}]$, $i < i' < j' < j$; hence, one cannot, for example, retrieve the k-NN set for each line segment in the query interval and then use only those to find the overall k-NNs.

5.1.6 Approximate Algorithm Using Line Simplification

The exact algorithm presented above may still require a large number of iterations, since it needs to compute the similarity between a large number of line segments. This is especially true when the query time interval becomes larger. In fact, in the worst case, i.e. when the bounds don't allow for early pruning, it needs to iterate over all the line segments of the trajectory within the query interval, in order to determine the exact value of the similarity.

An alternative approach to overcome this drawback is to approximate each trajectory with a simplified version. This introduces some error due to the approximation, but it can significantly reduce the number of line segments to be examined, thus allowing a trade-off between accuracy and efficiency. This approach works particularly well when the movement of an object is not characterized by frequent and abrupt changes in direction.

Several techniques exist in the literature for line simplification [SC06]. In our approach, we use the sleeve-fitting polyline simplification algorithm described in [ZS97], which has been shown to be time efficient while still achieving high accuracy. The basic idea of this technique is to cover the whole trajectory by fitting consecutive points into maximum sleeves, i.e. rectangular strips in 2D of width $2 \cdot \varepsilon$, where ε is an error tolerance parameter. Then, the center-line of each sleeve offers a one-segment approximation to the sub-trajectory of the original trajectory comprising all the consecutive points inside that sleeve. An example is shown in Figure 5.1, where a trajectory $<p_1, p_2, p_3, p_4, p_5, p_6, p_7>$ is simplified to $<p_1, p_5, p_7>$.

The approximate algorithm, outlined in Algorithm 6, firstly applies the simplification technique described above to produce a simplified version of the query trajectory (line 2). Subsequently, it only considers for each object the timestamps in the simplified version of the query trajectory (lines 3–7). Finally, it executes the exact algorithm on these simplified trajectories (line 8).

Notice that the line simplification step is applied only to the query trajectory and not to all examined trajectories. The reason for this is that, when comparing

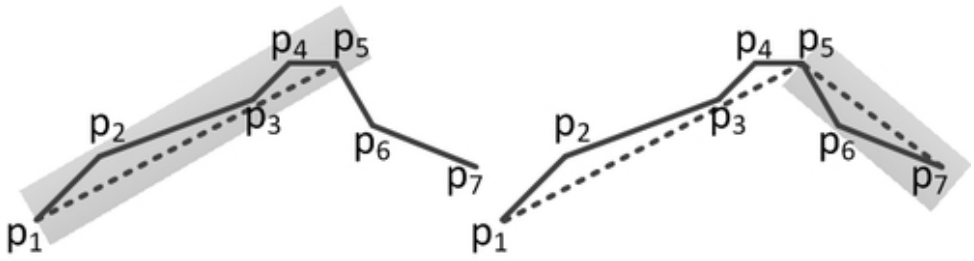


Figure 5.1: *Illustration of sleeve-fitting polyline simplification.*

ALGORITHM 6: Approximate computation of k-nearest moving neighbors using line simplification

Input: A database of trajectories D, a query trajectory Q, a query time interval $[t_s, t_e]$
Output: Approximate set of k-nearest moving neighbors D'_k

```
1  begin
2      Q' ← SleeveFittingApproximation(Q)
3      D' ← ∅
4      foreach P ∈ D do
5          P' ← P[t ∈ Q']
6          D' ← P'
7      end
8      D'_k ← ExactKNN(D', Q', [t_s, t_e])
9      return D'_k
10 end
```

two trajectories, we want to compare line segments between matching timestamps. Hence, we retain as reference the timestamps in the simplified version of the query. Instead, if the approximation was performed in both trajectories, the timestamps in the resulting simplified versions would not match (in the general case). This would incur an additional overhead for "filling in" the missing points in each trajectory in order to match each other's timestamps.

5.1.7 Approximate Algorithm Using Prior Probability Acceleration

Next, we examine a different direction for reducing the execution time of the exact k-nearest moving neighbors algorithm. The main idea is to adopt some basic aspects of Bayesian probability theory and to use prior probabilistic knowledge on trajectory data, in order to accelerate the approximation of the k-nearest neighbors. In Bayesian decision theory, the prior of an uncertain variable refers to the probability distribution that expresses the uncertainty about it before examining the actual data. It is a concept widely used in many classification and other machine learning problems.

To this end, we employ a pre-processing step to compute the parameters of

probability density functions of the trajectory line segments. Then, at query time, we use them to estimate the prior probability of a line segment of a moving object being close to the corresponding query line segment. The exact algorithm is adapted to use these priors in order to boost its convergence to an approximate set of k-NNs. Hence, this is a hybrid approach, combining a deterministic part, i.e. the similarity between line segments derived through a set of Euclidean distances between their endpoints, and a probabilistic part.

For the computation of the aforementioned prior probabilities, we need to make an assumption about the probabilistic model we will use, in order to estimate the degree of "similarity" (prior probability of a moving object being close to the query object) between moving objects. In this contribution, we have assumed that the distribution of line segments w.r.t. the corresponding in time line segment of a query trajectory, can be described by a Gaussian distribution. The Gaussian density function $g(x; \mu, \Sigma)$ of x, which is a d-dimensional data vector, with mean vector $\mu \in \mathbb{R}^d$ and a covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$, is defined as follows:

$$g(x; \mu, \Sigma) = (2\pi)^{-\frac{d}{2}} \times \det(\Sigma)^{-\frac{1}{2}} \times \exp\left(-\frac{1}{2}(x-\mu)^{\intercal}\Sigma^{-1}(x-\mu)\right) \tag{5.7}$$

In our case, we calculate bivariate Gaussian probability density functions, as we are dealing with 2-dimensional points. Hence, x is a 2×1 vector and Σ is a 2×2 positive definite matrix. In the pre-processing step, we compute the parameters μ and Σ of a Gaussian distribution for each line segment of a trajectory. More precisely, the computation is done on the midpoints of the line segments. The process is outlined in Algorithm 7. Let $\overline{L}$ denote the midpoint of a line segment L. For each line segment L between timestamps t_i and t_{i+1}, first the parameter μ is set to its midpoint (line 4). Then, the algorithm constructs a set M containing the midpoints of all other line segments of other trajectories between the same timestamps t_i and t_{i+1} (lines 5–8). Finally, the Gaussian covariance matrix Σ is calculated on μ and M (line 9). Notice that, due to the low dimensionality of our problem, this pre-processing procedure has very low execution time.

An example is illustrated in Figure 5.2. Figure 12 plots the midpoints of the line segments for a selected pair of timestamps t_i, t_{i+1}. Figure 12 shows the calculated Gaussian distribution of the midpoints of a selected pair of timestamps with the Gaussian mean set to the marked as square midpoint in Figure 12.

At query time, the process of finding the k-nearest neighbors follows the same steps as the exact algorithm outlined in Algorithm 5. The difference here lies in how the similarity bounds are updated when a line segment of a moving object is examined, i.e. in the method UpdateBounds. In this case, the prior probability of the examined line segment being close to the query line segment is calculated using Equation 5.7. Then, it is used as a scaling factor for the similarity bounds. The modified version of the method UpdateBounds is shown in Algorithm 8.

5.1.8 Experimentation

This section describes our experimental setup and presents the results of our evaluation. The performance of our algorithms is investigated on a set of three real-world

ALGORITHM 7: Computing line segment distributions

Input: A database of trajectories D
Output: Probability distribution parameters for each line segment

```
1  begin
2      foreach P ∈ D do
3          foreach L ∈ P do
4              L.μ ← L̄            /* the midpoint of L */
5              M ← ∅
6              foreach P′ ∈ D do
7                  M ← add L′     /* the line segment of P′ between
                   the same timestamps as L */
8              end
9              L.Σ ← CovEst(L.μ, M)
10         end
11     end
12 end
```

ALGORITHM 8: Modified *UpdateBounds* with priors

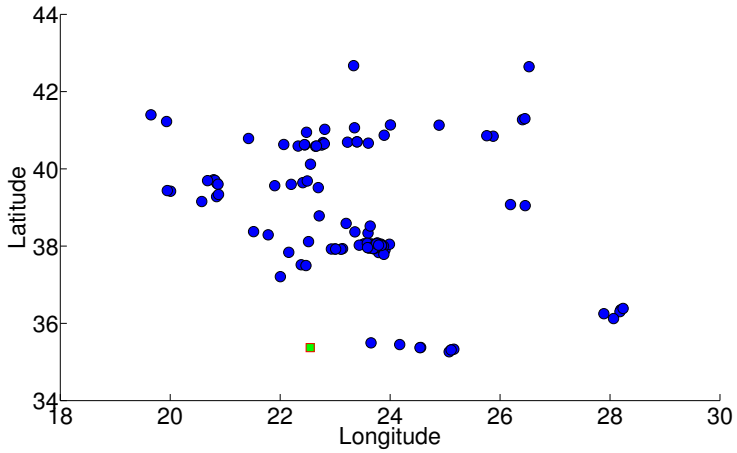
```
1  Function UpdateBounds(P, r, L_q^{t_p}, L_p^{t_p})
2      begin
3          prior ← g(L̄_p^{t_p}; L_q^{t_p}.μ, L_q^{t_p}.Σ)
4          sim_p^- ← prior × (sim_p^- + r/(e−s))
5          sim_p^+ ← prior × (sim_p^+ − (1−r)/(e−s))
6      end
```

trajectory datasets. We conduct a detailed experimental evaluation, measuring both the efficiency of the algorithms, in terms of execution time, and the accuracy of the approximate algorithms w.r.t. the exact algorithm, in terms of recall and Spearman distance. The purpose is to examine the effect of two parameters: the number k of nearest neighbors to be retrieved, and the width of the time interval indicated in the query. In the following, we present the datasets used, the evaluation measures, and the experimental results.

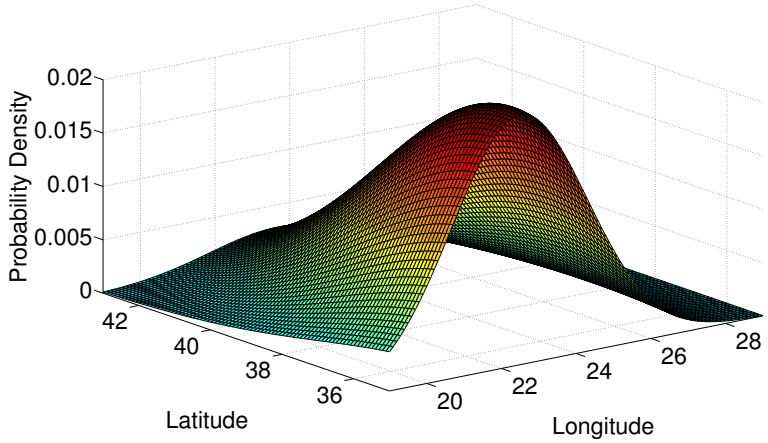
5.1.8.1 Datasets

We have conducted our experiments using three real-world datasets of trajectories, referred hereafter as T-drive, Ships, and Athens vehicles. These datasets cover a variety of cases regarding the shapes of trajectories. For example, in the T-drive dataset, the shape of the trajectories exhibits a relatively high regularity due to the grid-like structure of the underlying road network. At the other end, the Athens road network is highly irregular, resulting in a trajectory dataset with high heading variance. Finally, the ships trajectory dataset comprises relatively long trajectories with medium degree of variations.

Notice that a typical issue in trajectory datasets is the (often high) variation of



(a) Positions of line segment midpoints.



(b) Estimated distribution.

Figure 5.2: *Example of line segment midpoints with the associated distribution centered around a line segment midpoint (square).*

the sampling rate. This is a problem caused, for example, by low GPS signal strength or by the fact that the users often tend to switch on and off their GPS devices several times during the day (e.g., to save battery or due to privacy concerns). To overcome this problem, we perform a pre-processing of the datasets which involves two main actions: (a) we employ linear interpolation to create records for the location of each object for fixed time intervals (in particular, every 30 seconds); (b) whenever the timespan between two consecutive traces of an object exceeds a specified threshold (we used 120 seconds), we split the trajectory to two separate sub-trajectories.

Below we describe in more detail the characteristics of the aforementioned datasets.

5.1.8.2 T-Drive Trajectory Dataset

This dataset comprises GPS tracking data from taxis moving in the area of Beijing [YZXS11, YZZ+10]. We used a total of 1,023,924 trajectories from 569 taxis

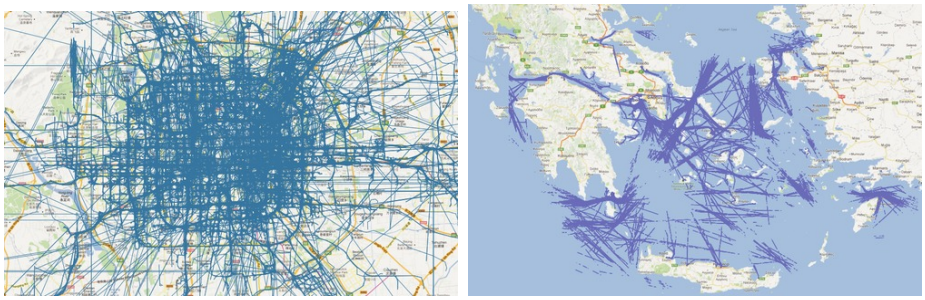
recorded in the period 2/2/2008 – 4/2/2008. On average, each trajectory comprises 3,016 line segments. A plot is shown in Figure 5.1.8.4.

5.1.8.3 Ship Trajectory Dataset

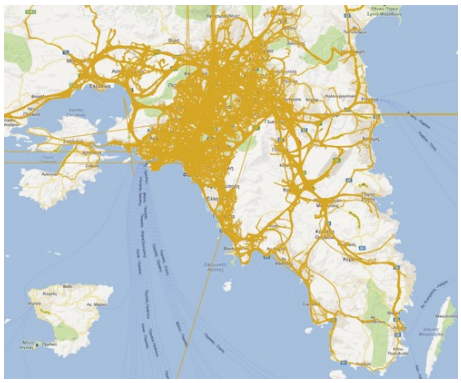
This dataset comprises GPS tracking data from ships moving in the Aegean Sea[2]. We used a total of 986,275 trajectories from 887 ships recorded in the period 31/12/2008 – 02/01/2009. On average, each trajectory comprises 1,100 line segments. A plot is shown in Figure 5.1.8.4.

5.1.8.4 Athens Vehicles Trajectory Dataset

This dataset comprises GPS tracking data from vehicles moving in the area of Athens, recorded in the context of the SimpleFleet project[3]. We have used a total of 667,421 trajectories from 2,497 vehicles recorded on 01/10/2012. On average, each trajectory comprises 156 line segments. A plot is shown in Figure 5.1.8.4.



(a) T-Drive (b) Ships



(c) Athens vehicles

Figure 5.3: *Plots of trajectory datasets used for the experimental evaluation.*

5.1.8.5 Evaluation Measures

The conducted experimental evaluation focuses on two aspects. The first is to assess the efficiency of the exact and the approximate algorithms by measuring the exe-

[2]http://www.chorochronos.org/?q=node/8
[3]http://www.simplefleet.eu/

cution time for the datasets described above. All the algorithms were implemented in Java and the experiments were executed on an Intel(R) Core(TM) i5-2400 CPU at 3.10GHz with 8GB of RAM, running Ubuntu 12.04 LTS. The trajectories were stored in a PostGIS database.

The second aspect is to examine the accuracy of the approximate algorithms w.r.t. the exact algorithm. A simple and most typically used measure for this purpose is recall. In information retrieval tasks, recall measures the portion of relevant items that have been retrieved by the algorithm. In our case, if D_k denotes the actual set of k-NNs and D'_k the one returned by an approximate algorithm, then

$$Recall = \frac{D_k \cap D'_k}{k} \tag{5.8}$$

Notice that, since both sets have the same size k, in this case this also corresponds to precision, which measures the portion of retrieved items that are relevant.

However, recall takes into consideration only the overlap of the two k-NN lists, ignoring their ordering. Although this may be sufficient for some mining tasks, there are also many cases where the ordering is important. To measure this, we use Spearman's footrule distance [KG90], which is often used for comparing different rankings. More specifically, we use the version proposed in [FKS03], which also deals with cases where each of the ranked lists may contain objects not appearing in the other. Using the same notation as above for D_k and D'_k, this measure is computed as:

$$F(D_k, D'_k) = \frac{\sum_{i \in D_k} |rank(i, D_k) - rank(i, D'_k)|}{(k+1) \times k} \tag{5.9}$$

where $rank(i, D) = |D|+1$ if $i \notin D$. Thus, $F(D_k, D'_k) = 0$ when the two ranked lists are identical, i.e. contain the same objects with the same order, while $F(D_k, D'_k) = 1$ when the two ranked lists contain completely different objects.

5.1.8.6 Results

In the following, we present the results of the evaluation. We measure the execution time and accuracy of the algorithms, examining the effect of two parameters: (i) the number k of nearest neighbors, and (ii) the width of the query time window. For the former, we vary k from 10 to 50 with step 10, using a default query window of 90 minutes; for the latter, we vary the width of the query window from 30 minutes to 150 with step 30, using as default $k = 30$.

Moreover, for the line simplification step used in Algorithm 6, we set the error parameter to $\varepsilon = 0.001$. The number of line segments in the simplified trajectories is in the order of 20% of the original ones for the T-drive dataset, 5% for the ships dataset, and 35% for the Athens dataset.

Each experiment has been repeated for 10 randomly selected queries, and the average results are reported.

5.1.8.7 Execution time

Figure 5.4 shows how the execution time of the three algorithms varies w.r.t. the number of k-nearest neighbors and the width of the query time interval, for the

T-Drive, Ships and Athens vehicles datasets respectively.

In all cases, the two approximate algorithms perform much faster than the exact algorithm, especially as the query time window becomes larger. This is justified by the fact that as the query time window grows, the exact algorithm needs to iterate over a larger number of line segments, while the approximate algorithms still terminate earlier, thus are affected less.

Both approximate algorithms exhibit very low execution times in all experiments, without being affected much by the various parameters. This especially holds for the approximate algorithm based on probabilistic priors, since scaling the similarity bounds with these priors leads to a faster convergence, i.e. earlier termination. The execution time of the approximate algorithm based on line simplification is higher due to the sleeve fitting simplification applied to approximate the query trajectory. The difference is higher for the ships and Athens vehicles datasets, where the trajectories are relatively more irregular compared to the T-drive dataset.

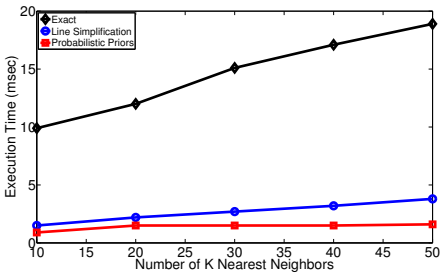
5.1.8.8 Accuracy

Figure 5.5 shows how the accuracy of the approximate algorithms varies w.r.t. the number of k-nearest neighbors and the width of the query time interval, for each of the three datasets.

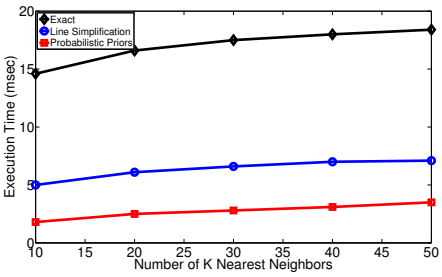
Overall, the approximate algorithm based on line simplification achieves quite high accuracy in all cases, with a recall exceeding 0.8 in most cases, and a Spearman distance below 0.2. In contrast, the accuracy of the approximate algorithm based on probabilistic priors varies in different cases. In particular, it appears to be less effective for the ships and Athens vehicles datasets, having low recall and high Spearman distance. However, its accuracy improves for the T-drive dataset, reaching a recall of 0.8.

This behavior is attributed to the following. As explained in Section 5.1.3, the similarity measure used for comparing trajectories takes into consideration both proximity (perpendicular and parallel distance) and direction (angle distance). When dealing with irregular trajectory data, we end up with distributions with high values of variance and low correlation between their variables (i.e., high probability of error). Scaling the similarity with such prior probabilities results in boosting the convergence of the similarity bounds with high probability of error. This means that trajectories starting with an erroneously estimated high similarity value (e.g. for their first few line segments) will converge very fast, introducing false positives to the final set of k-NNs.

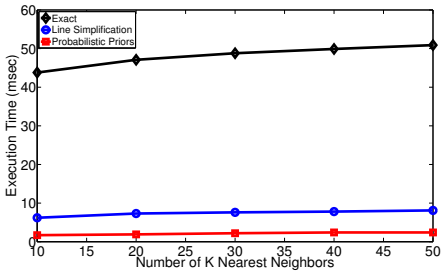
Consequently, the accuracy of the approximate algorithm based on probabilistic priors drops when dealing with cases where several line segments are located close to each other but are heading to different directions. This phenomenon appears less in the T-drive dataset. This dataset comprises trajectories of taxis moving in the road network of Beijing, which has a relatively regular, grid-like structure. In contrast, in the ships dataset, it is more frequent to have ships moving in the same area but in different directions, thus the accuracy of the algorithm reduces. Similarly, in the Athens vehicles dataset, this phenomenon is again prominent due to the highly irregular structure of the Athens road network, which explains the low accuracy in this case.



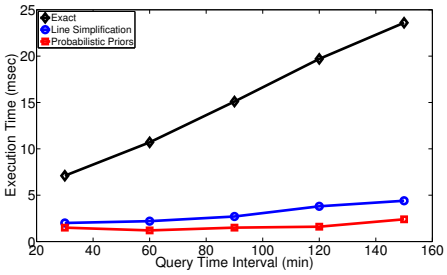
(a) T-Drive



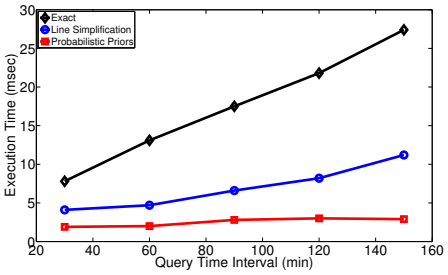
(b) Ships



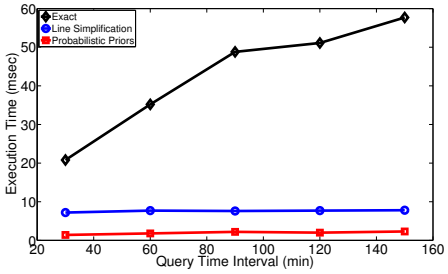
(c) Athens vehicles



(d) T-Drive

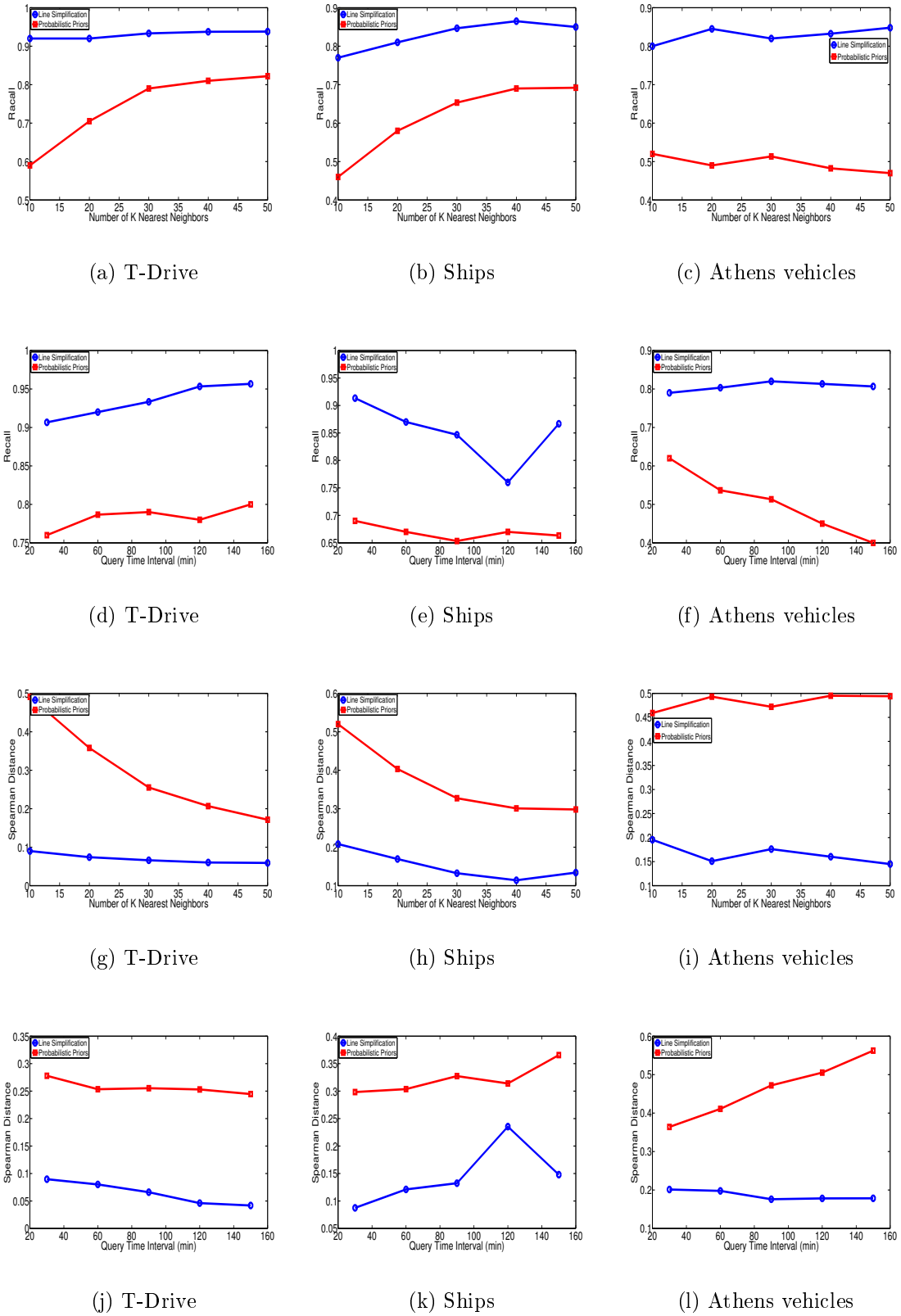


(e) Ships



(f) Athens vehicles

Figure 5.4: *Execution time vs k (a, b, c). Execution time vs query time interval (d, e, f).*



(a) T-Drive (b) Ships (c) Athens vehicles

(d) T-Drive (e) Ships (f) Athens vehicles

(g) T-Drive (h) Ships (i) Athens vehicles

(j) T-Drive (k) Ships (l) Athens vehicles

Figure 5.5: *Recall vs k (a, b, c) and query time interval (d, e, f). Spearman Distance vs k (g, h, i) and query time interval (j, k, l).*

5.2 Continuous Monitoring of KNN Trajectories

Analyzing tracking data of various types of moving objects is an interesting research problem with numerous real-world applications. Several works have focused on con-

tinuously monitoring the nearest neighbors of a moving object, while others have proposed similarity measures for finding similar trajectories in databases containing historical tracking data. In this work, we introduce the problem of continuously monitoring nearest trajectories. In contrast to other similar approaches, we are interested in monitoring moving objects taking into account at each timestamp not only their current positions but their recent trajectory in a defined time window. We first describe a generic baseline algorithm for this problem, which applies for any aggregate function used to compute trajectory distances between objects, and without any restrictions on the movement of the objects. Using this as a framework, we continue to derive an optimized algorithm for the cases where the distance between two moving objects in a time window is determined by their maximum or minimum distance in all contained timestamps. Furthermore, we propose additional optimizations for the case that an upper bound on the velocities of the objects exists. Finally, we evaluate the efficiency of our proposed algorithms by conducting experiments on three real-world datasets.

5.2.1 Preliminaries

The increasingly widespread use of GPS enabled devices and other positioning technologies has made possible the tracking and monitoring of various types of moving objects, such as cars, people or animals. This has consequently lead to the study of a broad range of queries in a multitude of settings and applications. Retrieving objects whose motion is "similar" to that of a target query object is one of the most basic and useful analytical queries. In the literature, two important types of such moving object similarity queries have been proposed, the *Nearest Neighbor* (NN) and the *Nearest Trajectory* (NT), also known as trajectory similarity, queries. Both types define a similarity (or equivalently a distance) metric between moving objects, and return the top-k most similar (or equivalently least distant) objects with respect to a specified moving query object. The distinguishing characteristic is the definition of the metric. Generally speaking, NN queries, e.g., [FGPT07b, GBX10b], are concerned with the distance between *individual locations* of moving objects, i.e., at some particular time instance, whereas NT queries, e.g., [PKM$^+$07a, SSV13] take into account the distance between the *trajectories* of moving objects, i.e., for the sequence of locations over a time interval.

Both query types have been extensively studied for *historical* data, which can be stored on disk and indexed by specialized data structures. To the best of our knowledge, however, only NN queries have been considered in a *continuous monitoring* setting, where new object locations continuously arrive, and the result must be accordingly updated. This work introduces and studies *Continuous Nearest Trajectory* (CNT) queries. Given a *trajectory distance*, i.e., a metric aggregating individual location distances within a specified time window, a CNT query continuously returns the set of k objects that have the smallest trajectory distance to a given query object. CNT queries are a natural extension of both continuous NN queries, in the sense that the recent trajectory (and not only the last location) of objects is considered, as well as of historical NT queries, in that the result is computed and maintained in real-time.

We note that existing approaches do not extend for CNT queries. This is obvious for methods designed for historical data, as they take advantage of specialized index

structures (which are not suitable for highly dynamic streaming data), and have the entire trajectory completely known upfront. Moreover, algorithms for continuous NN queries cannot be adapted for CNT. The main reason being that these methods assume that either the objects [BPPP08] or the query [YPK05, XMA05, MPH05] is stationary, and define *validity* or *influence spatial regions*, which guarantee that the result will not change as long as the query object remains inside the region, in the former case, or that objects do not cross the region, in the latter case. Note that the latter methods can handle moving queries but only by treating them as new queries. This means that previous computations are no longer useful and the result needs to be computed from scratch, a scenario which is only tolerable when the query object changes location infrequently. Therefore, a validity/influence region-based approach is not possible for CNT queries, where both the query object and the other objects move continuously and freely. However, we show that, in a specific setting (concerning the definition of the trajectory distance and assuming maximum velocities), it is possible to determine the minimum expected time when a moving object can influence the result.

There are some other works dealing with different types of continuous queries on moving objects, which however do not extend for CNT queries either. In a setting similar to ours, [BT08] defines a trajectory distance metric, and continuously computes the spatiotemporal trajectory join, i.e., determines pairs of objects whose trajectory distance does not exceed a given threshold. In other words, the underlying computation is answering a *range query* under a *hard threshold*, which is always easier to process as the search space is restricted. In contrast, the k-th trajectory distance in CNT queries is not known beforehand and can be arbitrarily high, making the methods of [BT08] inapplicable for CNT queries. Another work [VBT09b] proposes an online method to determine groups of objects that move close together, i.e., within a disk of a given radius. In their problem, only the distance between individual locations is taken into account and the threshold is also hard, making their ideas not suitable for CNT queries.

Given these observations, we propose a generic baseline algorithm, termed BSL, for processing CNT queries. This approach makes no assumptions regarding the underlying trajectory distance function or the movement of the objects. Thus, it serves as a framework for adapting and optimizing algorithms to more specific cases.

Building upon this, we derive an optimized algorithm, called XTR, for the cases where the trajectory distance between two moving objects is defined based on the extrema (maximum or minimum) of individual location distances. The maximum-defined CNT query establishes a distance guarantee that spans the time window within which trajectories are examined, and can be used, for example, to determine how far the nearest objects have strayed. The minimum-defined CNT query determines objects that have come close to the query at any time during the recent past, and can be thought of as a continuous NN query with "memory". On the other hand, using both the minimum and the maximum location distances, gives a more informative description of the movement of an object, as it determines the tightest annulus (donut) around the query that contains the object's trajectory.

Moreover, we study the aforementioned case when a global maximum velocity for the objects is known. This is a reasonable assumption given that most moving objects have upper bounds on their attainable velocities. For this particular setting, we introduce the HRZ algorithm, which computes distance bounds in order to

determine the earliest possible time, termed *horizon*, when an object may influence the result.

Our main contributions can be summarized as follows:

- We introduce and formally define the problem of continuously monitoring the objects with the k-nearest trajectories to a given query object, where trajectory distances take into consideration the objects' recent locations.
- We present a generic baseline algorithm (BSL) for the problem, which defines the types of events and operations needed for the computation.
- We propose the XTR algorithm optimized for the case when the trajectory distance is determined by the maximum or minimum individual location distance between objects. We also discuss some other related trajectory distance definitions.
- We present the HRZ algorithm that introduces further optimizations assuming that the moving objects have bounded velocities.
- We experimentally evaluate the proposed algorithms using real-world datasets.

5.2.2 Related Work

We discuss various types of queries for moving objects, distinguishing between NN variants in Section 5.2.2.1 and NT methods in Section 5.2.2.2.

5.2.2.1 NN Queries on Moving Objects

Given a (stationary) query object location and a set of (stationary) object locations, the k-Nearest Neighbor (NN) query retrieves the k objects which are closer to the query location. There are many different ways to extend the NN query for moving objects during some time interval, evident by the rich bibliography on the subject.

A first classification is based on where the interval of interest lies with respect to the current time. If it is in the past, the queries are termed *historical*, as they concern stored trajectory data. If the interval is placed in the future, the queries are further classified into *predictive*, when it can be assumed that objects move in a known manner (i.e., with constant velocity, or along a line) and thus their future locations can be extrapolated, or *monitoring*, when no assumptions are made on the moving patterns and thus location updates are issued. Processing historical and predictive NN queries is generally less challenging compared to monitoring queries, because the former essentially have at query time the entire trajectories of the moving objects.

The second classification is based on the semantic of the NN query during an interval. A *snapshot* NN query reports the objects that are closest to the query object at *any* time instance within the interval; e.g., find the object that comes closest to some location within the next 10 minutes. A *continuous* NN query reports the objects that are closest to the query object at *every* time instance within the interval; e.g., report the objects that were at some time closest to the query object during the past 10 minutes. Note that in the data stream literature, the term continuous (or long standing) query [BW01] refers to the case when the result of a query must be continuously updated as streaming tuples arrive; in the context of

NN queries, these requirements essentially correspond to the continuous monitoring NN query.

Regarding predictive queries, [KGT99] presents a dual plane method for predictive snapshot NN queries, in the case that all objects move in 1-D space, or are restricted to move within the same segment (i.e., road). [TP02] studies continuous predictive variants for various spatial queries, including NN, and describe a method to return the initial result and its validity period (i.e., the time at which the result will change). [GZC+11] studies continuous predictive NN queries assuming that only the query is moving along a line, while all other objects are stationary. [ISS03] and [BJKx06] also deal with continuous predictive NN queries, but they are able to handle updates on the motion patterns of objects, without computing the result from scratch.

For continuous monitoring NN queries, [SR01] and [BPPP08] handle the case when only the query object is moving. The former retrieves $m > k$ nearest neighbors hoping that the result at a future time is among these m objects, provided that the query does not move much. The latter returns a Voronoi-based validity region such that the result does not change as long as the query remains within the region. [YPK05], [XMA05] and [MPH05] present incremental grid-based methods for general continuous monitoring NN queries, i.e., when all objects move in a non-predictive manner; the last two works feature shared execution techniques to handle multiple NN queries.

In the case of historical trajectory data, R-tree based trajectory indices (e.g., 3D R-tree [VTS98], TB-tree [PJT00b]) are typically used to expedite the NN query processing. [GLC+06] handles historical snapshot NN queries, while [FGPT07b], [GBX10b] process historical continuous NN queries.

Another line of work concerns NN queries over uncertain data. For example, [TTD+09] processes continuous monitoring NN queries for objects with uncertain locations. [HLL08] handles continuous predictive NN queries with updates for objects with uncertain locations and speeds. [NZE+13] deal with historical snapshot and continuous NN queries for objects with uncertain locations.

Finally, there has been some interest on identifying groups of moving objects, such as moving object clusters [KMB05b], flocks [GvK06a, VBT09b], convoys [JYZ+08b], and swarms [LDHK10b]. Generally speaking, these groups consist of objects that are close to each other (e.g., within a disk of a given radius) at each time instant. These methods however cannot be used for processing CNT queries.

5.2.2.2 Nearest Trajectories

There exist many approaches for defining distance (or similarity) metrics for trajectories. All of them also propose methods to identify the most similar trajectory to a given query trajectory, which can be extended to retrieve the top-k similar ones, but their techniques only operate on historical data. A useful survey on the topic is included in [PKM+07a].

While the Euclidean distance (or some other L_p norm) is typically used to quantify closeness of two locations, the extension for the case of multiple locations within trajectories is not straightforward. In addition, a trajectory distance must take into account the temporal aspect of the locations. [YAS03b] defines the trajectory distance as the L_2 norm of individual Euclidean location distances, after re-sampling

the trajectories to account for different reporting intervals. [LS05] ignores the temporal dimension and defines spatial trajectory distance as the average of the Euclidean distances computed between a location in one trajectory and its closest location in the other (termed the one way distance).

The previous trajectory distances can be computed in linear time with respect to the trajectory length. On the other hand, there exist more complex metrics, inspired from sequence similarity measures, that require quadratic time. [VGK02a] uses the Longest Common Subsequence (LCSS) similarity measure, an edit distance variant, that allows the matching of locations that are close in space at different time instants, provided that they are not far in time, and also allows for locations to be unmatched, e.g., accounting thus for location imprecisions or small deviations. In a similar manner, [COO05b] defines the Edit Distance on Real Sequence (EDR) that captures the minimum number of edit operations (insert, delete, replace locations) necessary to transform one trajectory into the other.

An approach for finding historical top-k similar trajectories is presented in [SSV13]. The basic algorithm prioritizes object examination aiming to avoid distance computations for objects not in the result. In addition, approximate techniques are also presented.

Another related problem is trajectory clustering, where the goal is to group trajectories based on a trajectory distance metric. For this problem, however, the basic underlying operation is typically a range query (retrieve trajectories within a given distance threshold) rather than a top-k similarity query. For example, in [LHW07b] the goal is to partition historical trajectories into sub-trajectories and then group them to construct dense clusters according to a metric that composes a perpendicular, a parallel, and an angle distance.

To the best of our knowledge, continuous monitoring of top-k similar trajectories has not been addressed in the past. The only work that handles continuous monitoring of a trajectory defined query is [BT08], which deals with spatiotemporal trajectory joins. The underlying trajectory distance metric is the maximum among all Euclidean location distances, and the goal is to find pairs of trajectories that are within a given trajectory distance threshold. That is the core query is a range rather than a top-k similarity query. Therefore, their approach is not applicable to our problem.

5.2.3 Contribution

5.2.3.1 Problem Definition

Consider a set O of moving objects, whose locations are continuously monitored and reported at fixed discrete times, called *timestamps*. Location updates have the form $< o, t, x, y >$, meaning that object o at timestamp t is at location $o[t] = (x, y)$. We assume that updates always arrive in increasing order of their timestamps, but we do not assume that for each timestamp updates are received for all objects.

We denote as $\mathcal{T}(o)$ the set of timestamps at which updates for o were received. For simplicity and without loss of generality we assume that for any timestamp t' for which no update for o was received, i.e. $t' \notin \mathcal{T}(o)$, the location of o is the same as its last reported location, i.e. $o[t'] = o[t]$, where $t \in \mathcal{T}(o)$ is the latest timestamp before t'. Essentially, this corresponds to assuming that object o has not moved

during time $[t', t]$; making other assumptions can also be handled accordingly, e.g., by issuing artificial updates for the objects based on inferred locations.

Definition 5.2.1. *The* location distance *between two objects o and o' at timestamp t is given by the Euclidean metric, i.e.,*

$$d(o, o', t) = \sqrt{(x - x')^2 + (y - y')^2}.$$

where $o[t] = (x, y)$ and $o'[t] = (x', y')$ are the respective (reported or extrapolated) locations of o and o' at t.

The above definition measures the distance between two objects at a single timestamp. However, we are interested in comparing the recent trajectories of the objects, hence their distances over a series of consecutive timestamps within a specified time window. For this purpose, we introduce the following definition.

Definition 5.2.2. *Given two objects o and o', a time window w, and an aggregate function $\mathcal{G}$, the* trajectory distance *of o and o' is defined by applying $\mathcal{G}$ on the location distances of o and o' at each timestamp within the time window of length w ending at timestamp t:*

$$D(o, o', t, w, \mathcal{G}) = \mathcal{G}_{\tau \in [t-w, t]} d(o, o', \tau).$$

Function $\mathcal{G}$ can be any aggregate function, e.g., minimum, maximum, average, among others.

We now formally define the problem of continuously reporting the objects with the k-nearest trajectories to a moving query object.

Problem Statement. The *Continuous Nearest Trajectory* (CNT) query $< O, q, \mathcal{T}, k, w, \mathcal{G} >$, where O is a set of moving objects, $q \in O$ a query object, and $\mathcal{T}$ a series of consecutive future timestamps, returns for each timestamp $t \in \mathcal{T}$ the k objects in O that have the smallest trajectory distance to q w.r.t. the time window w and the aggregate function $\mathcal{G}$, i.e., $\forall t \in \mathcal{T}$ it returns a subset $O_k \subseteq O$ of size k, such that $\forall o \in O_k, o' \in O \setminus O_k$:

$$D(q, o, t, w, \mathcal{G}) \leq D(q, o', t, w, \mathcal{G}).$$

5.2.3.2 Baseline for CNT

We first describe a generic *baseline* (BSL) method for answering continuous nearest trajectory queries. BSL operates under *any* aggregate function $\mathcal{G}$ and follows an event driven process, where the events to be handled are specified below:

- *Query location updates* ($QUpd$). This is an update $< q, t, x, y >$ to the location of the query object q, specifying its new location (x, y) for the current timestamp t. This may result in changes in the trajectory distances of the objects, and subsequently changes in the current set of nearest trajectories (NTs). When objects are allowed to move arbitrarily, their new distances to the new query location have to be computed, and the new aggregate distances and NTs have to be evaluated.

- *Object location updates* (*OUpd*). This is an update $< o, t, x, y >$ to the location of an object o, specifying its new location (x, y) for the current timestamp t. As a result, the current location distance of the object to the query has to be evaluated, which may affect its trajectory distance within the window w. If this changes, it may in turn affect the inclusion or not of the object in the result set. In addition, the system needs to remember to purge this location distance when it becomes obsolete, i.e., concerns a location outside the window. Therefore, it generates a corresponding expiration event that will be triggered at timestamp $t + w$ as described next.

- *Object distance expiration* (*OExp*). Unlike *QUpd* and *OUpd*, which are events received by the external environment, *OExp* events are generated and triggered by the system as part of handling *OUpd* events. *OExp* events have the form $< o, t >$, and mean that a location distance for object o is set to expire at timestamp t (this location distance was computed for a location update received at timestamp $t - w$). Similarly to a location update, such a removal may affect the trajectory distance of the object, and consequently its inclusion in the set of NTs.

In the following, we describe in detail how BSL handles the above events to evaluate a CNT query.

BSL makes use of the following in-memory data structures. For the query, it only stores its latest location *q.loc*. For each object, it stores its latest location *o.loc*, as well as a list *o.hist* of location distances and their corresponding timestamps, ordered by time. In addition, it uses an event queue Q to store and process *OExp* location distance expiration events, i.e., for removing distances for timestamps outside the time window w. An *OExp* event $< o, t >$ means that at time t, BSL needs to purge an expired distance for object o. This is the least recent location distance in the list *o.hist*. Events in Q are inserted and processed in a FIFO manner, i.e. they are ordered by time. Finally, BSL maintains a results list R of size k, where each entry corresponds to an object and its trajectory distance, updated at every timestamp. Any object that does not appear in the list at time t has trajectory distance not less than the largest trajectory distance in R.

Algorithm 9 shows the pseudocode for BSL. Since this is a continuous query, BSL executes in a loop for every timestamp $t \in \mathcal{T}$ (line 1), i.e. as long as the query is standing, and at each iteration it reports the current result set R (line 17). The input at each timestamp is the set of *QUpd*, *OUpd*, and *OExp* events that have been received for processing. Based on these events, BSL determines the set of *affected objects* O_A that require processing at this timestamp (line 2). Note that the set O_A contains not only objects that have received location updates, but also objects for which an expiration event was triggered, or, in the case of a query update, all objects.

If the query object has moved to a new location, then *q.loc* is updated and new object distances need to be computed (lines 3–5). Otherwise, only those objects for which an update or expiration happened are marked for processing (lines 6–8). Subsequently, each affected object $o \in O_A$ is processed (lines 9–17). First, the procedure `BSL_ProcessObject` is invoked (line 10), which updates *o.loc* and *o.hist* accordingly, and recomputes the object's location distance and trajectory distance (see Algorithm 10 below). Then, BSL checks if the returned trajectory distance has

ALGORITHM 9: BSL

```
1  foreach t ∈ 𝒯 do
2      O_A ← ∅ // the set of affected objects at t
3      if QUpd then
4          q.loc ← (x_q, y_q) // update q's current location
5          O_A ← O // mark all objects for processing
6      else
7          foreach OUpd and OExp do
8              O_A ← O_A ∪ o // add the referred object in O_A
9          end
10     end
11     foreach o ∈ O_A do
12         D_o ← BSL_ProcessObject(o)
13         if D_o has changed then
14             if o in the results R then
15                 update o's entry in R
16             else if D_o smaller than the trajectory distance of R's last
               entry then
17                 delete R's last entry
18                 insert an entry for o in R
19             end
20         end
21     end
22     report t, R
23 end
```

changed (line 11). If so, then the result set R may need updating. In particular, if o was in the result, then its entry in R must be updated (lines 12–13) with the new trajectory distance. Otherwise, if the new trajectory distance is smaller than any trajectory distance in R, this means that o should be (tentatively) inserted in R, evicting the last entry (lines 14–16).

We next describe the procedure `BSL_ProcessObject`, shown in Algorithm 10, in more detail. If an expiration event has occurred for o, then the expired location distance is removed from $o.hist$ (lines 1–2). If the object's location has changed, $o.loc$ is updated (line 4). The new location distance of o is computed and added to the history (lines 5–6). Moreover a corresponding expiration event is added in Q (line 7). Finally, the new trajectory distance for o is computed and returned (lines 8–9).

5.2.3.3 Extrema-defined CNT

In the following, we assume that the aggregate function $\mathcal{G}$ defining the trajectory distance is max or min over location distances, or, more generally, any other function taking as input only the extrema (max, min) location distances. In these instances, the trajectory distance is determined by one (or two) location distances within the time window. Note, however, that these location distances may change over time, as new locations arrive and old ones expire. Nonetheless, we show that processing

ALGORITHM 10: BSL_ProcessObject

```
1  if OExp event for o was triggered then
2  |  remove the expired location distance from o.hist
3  end
4  if QUpd or OUpd event for o was received then
5  |  update o.loc, if changed
6  |  compute new location distance d
7  |  add d to o.hist
8  |  create OExp event for o at timestamp t + w
9  end
10 update trajectory distance D_o
11 return D_o
```

of extrema-defined CNT queries can be streamlined. We first start our discussion considering the case of the max function; the case of min can be handled in a similar manner, and is hence omitted. We then discuss the necessary changes to process CNT queries for any extrema-defined aggregate function.

When the aggregate function $\mathcal{G}$ is max, the trajectory distance of an object o is determined by the largest location distance within the time window w. In that case, we show that it is possible to discard some location distances which cannot influence the trajectory distance during their lifespan. Based on this observation, we describe the *Extrema* (XTR) algorithm, which is based on the BSL framework but reduces the number of location distances stored per object, and, consequently, the number of events generated and processed. The key observation of XTR is captured by the following lemma.

Lemma 5.2.1. *Given an object o, where $d < d'$ are two location distances at timestamps $t < t'$ for $t' - t \leq w$, the location distance d does not contribute to the trajectory distance of o for any timestamp after t'.*

Proof. Location distance d is valid, i.e., may contribute to the trajectory distance, during its lifespan ending at timestamp $t + w$. During the time interval $[t', t + w]$, location distance d' is also valid and greater, and thus dominates d. As a result, the trajectory distance, i.e., the maximum location distance, must be at least $d' > d$. □

The XTR algorithm uses the same data structures and variables as BSL and performs the same main operations described in Algorithm 9. However, XTR differs from BSL in the way it processes objects. In particular, we discern the following main differences. First, XTR only keeps the non-dominated location distances in *o.hist*, as Lemma 5.2.1 suggests. Second, at any time t, the event queue Q contains only a single entry per object o, and its semantics can be viewed differently: it now schedules trajectory distance recomputations rather than location expirations. By purging a priori those earlier location distances that are smaller than d, XTR avoids unnecessary triggering of the corresponding *OExp* events, thus avoiding unnecessary processing of objects whose trajectory distance cannot yet change.

The processing for an object o in XTR is handled by the procedure `XTR_ProcessObject` outlined in Algorithm 11. Its first tasks, removing expired location distances, updating the object's location, and recomputing the object's location distance to the

query, are identical to BSL's (lines 1–6). In addition, based on Lemma 5.2.1, XTR removes any location distances less than d from $o.hist$ (line 7). Let t' be the earliest timestamp that remains (line 8). XTR inserts in Q an event to expire the location distance at t', if no event for o in Q already exists, otherwise it resets the scheduled time of the existing event (lines 9–12). This event essentially schedules the next trajectory distance recomputation necessary for o (assuming that the trajectory distance is not affected by newer location updates until then). Finally, the trajectory distance is set to the earliest location distance and returned (line 13).

ALGORITHM 11: XTR_ProcessObject

```
1  if OExp event for o was triggered then
2  |  remove the expired location distance from o.hist
3  end
4  if QUpd or OUpd event for o was received then
5  |  update o.loc, if changed
6  |  compute new location distance d
7  |  add d to o.hist
8  |  remove from o.hist all location distances less than d
9  |  t' ← the earliest timestamp in o.hist
10 |  if Q contains OExp event for o then
11 |  |  update OExp's time to t' + w
12 |  else
13 |  |  insert in Q the event < o, t' + w >
14 |  end
15 end
16 return D_o ← earliest location distance in o.hist
```

We now discuss the general case where the aggregate function $\mathcal{G}$ is some function over the extrema (min and max) location distances. One example of such a function is the average of the minimum and maximum location distances recorded for an object within the current time window. Recall that Lemma 5.2.1 identifies location distances which are irrelevant for the max case; an analogous lemma holds for the min case. Therefore, when both the max and the min location distance contribute to the trajectory distance, we can discard location distances which are irrelevant for both extrema cases. Following this observation, we propose the following changes to the `XTR_ProcessObject` algorithm. For each object o, we maintain its minimum and maximum location distances for the current time window, denoted as $o.min$ and $o.max$, respectively, i.e. $o.min = \min\{o.hist\}$ and $o.max = \max\{o.hist\}$. In addition, we keep a time marker t_m which is the earliest timestamp of either $o.min$ or $o.max$. In the event queue Q, we still need to keep only one entry for each object o, set to $< o, t_m + w >$, to trigger a reevaluation of its trajectory distance when either $o.min$ or $o.max$ expires. Moreover, the early removal of unnecessary entries in $o.hist$ is now done as follows. When $o.min$ or $o.max$ changes, and t_m is set accordingly, we remove all entries from $o.hist$ with timestamp earlier than t_m. The reason for this is that for any location distance d with timestamp $t < t_m$ it holds that $o.min < d < o.max$ (otherwise, d would be the current min or max) and d cannot become a future $o.min$ or $o.max$ since it expires before them.

5.2.3.4 Exploiting Bounded Velocities for Extrema-defined CNT

This section considers extrema-defined trajectory distances and assumes that there exists a global upper bound v_{max} on the velocity of a moving object[4]. Under this realistic assumption, we show that it is possible to derive a more efficient algorithm than XTR for processing CNT queries. The proposed *Horizon* (HRZ) algorithm takes advantage of the velocity bound to further reduce the number of location updates that need to be processed. Similar to Section 5.2.3.3, we assume that the trajectory distance is the maximum location distance within the time window; the case of min is similar, while the more general case of extrema-defined functions can be handled in a straightforward manner.

The basic idea behind HRZ is the following. For ease of exposition, assume $k = 1$ and consider two objects o and o'. Let $\underline{D}[t]$ and $\overline{D}'[t]$ denote, respectively, a lower and an upper bound on the trajectory distances of o and o' to the query q at time t. Clearly, if $\underline{D}[t] > \overline{D}'[t]$ for any timestamp t within a time interval, then o cannot be in the result during that interval. Hence, in Section 5.2.3.5, we derive lower and upper bounds on trajectory distances. Then, in Section 5.2.3.6, we discuss the computation of the time horizon, which determines a time interval during which a particular object may not be a result. Finally, in Section 5.2.3.7, we put our ideas together and present the HRZ algorithm.

5.2.3.5 Bounds on Trajectory Distances

Let t be the current timestamp, and consider an object o for which the most recent location distance is d received at timestamp $t_d \leq t$, and its current trajectory distance is $D \geq d$, valid since timestamp $t_D \leq t_d$. A lower bound for the trajectory distance of o at any future timestamp $t' > t$ can be computed assuming that object o moves at maximum velocity v_{max} towards the query q, while q also moves at maximum velocity v_{max} towards o. As a result, since the last known update at t_d, the location distance of o to q is decreasing at a maximum rate of $2v_{max}$. Notice however that this will affect its trajectory distance only after both D and d have expired. The trajectory distance in this setting is clearly a lower bound for the trajectory distance of o for any possible motion of o and q. Thus, we derive the following lemma.

Lemma 5.2.2. *Given an object o at current timestamp t, with latest location distance d at timestamp $t_d \leq t$ and current trajectory distance $D \geq d$ valid since $t_D \leq t_d$, its trajectory distance for any future timestamp $t' \geq t$ is lower bounded by the function:*

$$\underline{D_o}[t'] = \begin{cases} D & \text{if } t \leq t' \leq t_D + w \\ d & \text{if } t_D + w < t' \leq t_d + w \\ \max\{d - 2v_{max} \cdot (t' - t_d), 0\} & \text{if } t' > t_d + w. \end{cases}$$

Proof. First, observe that at time t the history of the object (i.e., during the time interval $[t - w, t]$) certainly contains a location distance with value D at time t_D (determining the current trajectory distance) and another with value d at time t_d. It

[4]Note that the extension to differing maximum velocities across objects is straightforward and thus omitted.

may also contain other location distances, which however must have values between d and D. Consequently, it is easy to see that the lemma holds for the first two clauses.

Regarding the third clause, we need to show that for any future timestamp $t' > t$, the lower bound holds. Consider the location distances valid during the future time window $[t' - w, t']$; recall that location distance d is no longer valid. Let d_m be the largest valid location distance with timestamp $t_m \in [t' - w, t']$. Therefore, the trajectory distance at time t' is defined as d_m. Due to the bound on the velocity of objects, it holds that any object, o or the query q, from timestamp t_d (of o's known location update in the past) up to timestamp t_m cannot have traveled a distance greater than $v_{max} \cdot (t_m - t_d)$. As a result, the location distance of o cannot have decreased more than $2v_{max} \cdot (t_m - t_d)$ (but not become less than zero), which is the case that o and q travel towards one another (and travel together once they reach each other). Therefore, the location distance at t_m cannot be less than $d_m \geq d - 2v_{max} \cdot (t_m - t_d)$, and is also greater than zero. Since $t_m \leq t'$, the lower bound holds. □

In a similar way, we can also derive an upper bound for the trajectory distance of o in a future timestamp t'. This can be computed assuming that object o moves at maximum velocity v_{max} away from the query q, while also q moves at maximum velocity v_{max} away from o. As a result, since the last known update at t_d, the location distance of o to q is increasing at a rate of $2v_{max}$. Again, any updates will come into effect only as long as there exists no previous value that is still valid and greater. The trajectory distance in this setting is clearly an upper bound for the trajectory distance of o for any possible motion of o and q. Thus, we derive the following lemma.

Lemma 5.2.3. *Given an object o at timestamp t, with latest location distance d at timestamp $t_d \leq t$ and current trajectory distance $D \geq d$ valid since $t_D \leq t_d$, its trajectory distance for any future timestamp $t' \geq t$ is upper bounded by the function:*

$$\overline{D_o}[t'] = \begin{cases} \max\{D, d + 2v_{max} \cdot (t' - t_d)\} & \text{if } t \leq t' \leq t_D + w \\ d + 2v_{max} \cdot (t' - t_d) & \text{if } t' > t_D + w. \end{cases}$$

Proof. Consider the first clause, and a timestamp $t' \in [t, t_D + w]$; the corresponding time window is $[t' - w, t']$ and D is still valid. Let d_m denote the largest valid location distance with timestamp $t_m \in [t' - w, t']$. Trivially, if d_m is D, the upper bound holds. Assume otherwise, i.e., $d_m > D$. Using similar reasoning as in Lemma 5.2.2, the location distance of o from timestamp t_d up to timestamp t_m cannot have increased more than $2v_{max} \cdot (t_m - t_d)$. Therefore, $d_m \leq d + 2v_{max} \cdot (t_m - t_d)$, and the upper bound also holds for this case because $t_m \leq t'$. The second clause is proved in a similar way, given that D has now expired. □

5.2.3.6 Time Horizon of Objects

We now proceed to derive the minimum time required for an object $o \notin R$ to enter the result set. We refer to this as the *time horizon* of an object, corresponding to the earliest time for which the object's trajectory distance may become equal to (or less than) the trajectory distance of some object in R. Using Lemmas 5.2.2 and 5.2.3, the time horizon is formally defined as follows.

Definition 5.2.3. *Given the current result set R at timestamp t, the* time horizon *t_h for an object $o \notin R$ is defined as the earliest possible time that the trajectory distance of o becomes lower than that of any object in R, i.e.:*

$$t_h = \min\{t' \geq t \mid \exists o' \in R : \underline{D_o}[t'] \leq \overline{D_{o'}}[t']\}$$

An important remark regarding the previous definition is that it does not suffice to just consider the trajectory distance upper bound of the k-th object in R. As location updates may not occur at all timestamps, it is possible for two objects $o_i, o_j \in R$ with trajectory distances $D_i < D_j$ to have at some future timestamp t' upper trajectory bounds such that $\overline{D_i}[t'] > \overline{D_j}[t']$. This can occur, for example, when the objects' last location distances and timestamps satisfy the conditions $d_i > d_j$ and $t_i < t_j$.

As a result, computing the time horizon for an object requires considering the trajectory distance upper bounds for all objects in R, which is time consuming given that the time horizon needs to be computed at each timestamp for each affected object not in the result set. We thus propose an alternative method for determining the time horizon. The key idea is the following lemma, which derives a single upper bound on the trajectory distance of any object in the result set R.

Lemma 5.2.4. *Consider a set of objects R at current timestamp t, where, for the i-th object, d^i denotes its latest location distance at timestamp t_d^i and $D^i \geq d^i$ denotes its current trajectory distance valid since timestamp $t_D^i \leq t_d^i$. Define object $o^+ \in R$ to be the one with the largest trajectory distance, and object $o^* \in R$ to be the one that can have the largest possible location distance at current timestamp t, i.e.,*

$$o^+ = \underset{o_i \in R}{\operatorname{argmax}} D^i \quad \textit{and} \quad o^* = \underset{o_i \in R}{\operatorname{argmax}} \left(d^i + 2v_{max} \cdot (t - t_d^i)\right).$$

Then, the trajectory distance of any object in R for any future timestamp $t' \geq t$ is upper bounded by the function:

$$\overline{D_R}[t'] = \begin{cases} \max\{D^+, d^* + 2v_{max} \cdot (t' - t^*)\} & \text{if } \; t \leq t' \leq t + w \\ d^* + 2v_{max} \cdot (t' - t^*) & \text{if } \; t' > t + w, \end{cases}$$

where D^+ is the trajectory distance of o^+, and d^* is the latest location distance of o^* computed at timestamp t^*.

Proof. It suffices to show that the upper bound on the trajectory distance of each object in R according to Lemma 5.2.3 is always (i.e., for any $t' > t$) not greater than the upper bound provided by this lemma. Consider an object $o_i \in R$ and its trajectory distance upper bound:

$$\overline{D^i}[t'] = \begin{cases} \max\{D^i, d^i + 2v_{max} \cdot (t' - t_d^i)\} & \text{if } \; t \leq t' \leq t_D^i + w \\ d^i + 2v_{max} \cdot (t' - t_d^i) & \text{if } \; t' > t_D^i + w. \end{cases}$$

First note that $t_D^i < t$, and consider a future timestamp t' during the time interval $[t, t_D^i + w]$. Comparing the first clause of the two bounds, we can see that $D^+ \geq D^i$ from the definition of object o^+. On the other hand, from the definition of o^* we derive that $d^* + 2v_{max} \cdot (t - t^*) \geq d^i + 2v_{max} \cdot (t - t_d^i)$. Adding $2v_{max} \cdot (t' - t)$ to both sides of the inequality, we derive that the lemma holds.

Next, consider a future timestamp t' during the time interval $[t_D^i + w, t + w]$, and compare the second clause of $\overline{D^i}[t']$ to the first clause of $\overline{D_R}[t']$. With similar reasoning as before, we have that $d^* + 2v_{max} \cdot (t' - t^*) \geq d^i + 2v_{max} \cdot (t - t_d^i)$, and since the first clause of $\overline{D_R}[t']$ is always greater than the left-hand side of the inequality, the lemma holds.

In the case of a future timestamp $t' > t + w$, when the second clauses of the bounds apply, it is easy to see, using similar reasoning as before, that the lemma holds. □

Using the bound on the trajectory distance of any object in R, it is possible to efficiently compute a timestamp that never overestimates the time horizon, as the next lemma suggests.

Lemma 5.2.5. *Given the current result set R at timestamp t, the time horizon t_h for an object $o \notin R$ is not less than the following value:*

$$t_h \geq \min\{t' \geq t \mid \underline{D_o}[t'] \leq \overline{D_R}[t']\}.$$

Proof. Denote as A the set from Definition 5.2.3, i.e., $A = \{t' \geq t \mid \exists o' \in R : \underline{D_o}[t'] \leq \overline{D_{o'}}[t']\}$, and as B the set from this lemma, i.e., $B = \{t' \geq t \mid \underline{D_o}[t'] \leq \overline{D_R}[t']\}$. We claim that $B \subseteq A$ to prove the lemma. Since it holds that $\overline{D_R}[t'] \geq \overline{D_{o'}}[t']$ for any $o' \in R$ from Lemma 5.2.4, the condition of set B is harder to satisfy than that of A, and thus the claim $B \subseteq A$ holds. □

Lemma 5.2.5 suggests that we can compute, in constant time, a timestamp not greater than the time horizon as the solution of the equation $\underline{D_o}[t'] = \overline{D_R}[t']$. Henceforth, to simplify the presentation of HRZ, whenever we refer to the time horizon t_h or its computation, we mean the solution of this equation instead of Definition 5.2.3.

5.2.3.7 The HRZ Algorithm

Having a method to compute the time horizon of an object, we next detail the HRZ algorithm, highlighting its differences with respect to XTR. The data structures and variables that HRZ uses are as in XTR, with the exception that for each object HRZ additionally stores its time horizon t_h indicating the time after which the object may appear in the result set R. The computation of t_h is based on Lemma 5.2.5. The HRZ algorithm takes advantage of the time horizon to reduce the number of events processed as follows. At any timestamp before $t_h - w$, HRZ ignores updates for the particular object. During the time interval $[t_h - w, t_h]$, HRZ only stores the locations and location distances, since these are necessary to compute the trajectory distance at time t_h. However, it does not compute the trajectory distance, since it is guaranteed to be greater than those in R, and it does not add any events in Q. After the time horizon t_h, HRZ operates similar to XTR.

Algorithm 12 shows the pseudocode for HRZ. The main difference from BSL and XTR is that it handles the processing of affected objects in two phases. In the first phase (lines 9–11), HRZ considers only objects that are in R, i.e., objects that were reported as results in the previous timestamp. For these objects, the processing (handled by HRZ_ProcessObject) is essentially identical to XTR, as we later explain. Once processing is completed, the object's entry in R is updated if its trajectory distance changed.

ALGORITHM 12: HRZ

```
1  foreach t ∈ 𝒯 do
2      O_A ← ∅ // the set of objects marked for processing at t
3      if QUpd then
4          q.loc ← (x_q, y_q) // update q's current location
5          O_A ← O // mark all objects for processing
6      else
7          foreach OUpd and OExp do
8              O_A ← O_A ∪ o // add the referred object in O_A
9          end
10     end
11     foreach o ∈ O_A ∩ R do
12         D_o ← HRZ_ProcessObject(o)
13         update o's entry in R
14     end
15     identify objects o⁺ and o* in R // from Lemma 5.2.4
16     foreach o ∈ O_A \ R do
17         if t < o.t_h − w then continue
18         D_o ← HRZ_ProcessObject(o)
19         if D_o smaller than the trajectory distance of R's last entry
           then
20             delete R's last entry
21             insert an entry for o in R
22         end
23     end
24     report t, R
25 end
```

Between the first and second phase, HRZ scans all objects in R, and determines objects o^+ and o^* as defined in Lemma 5.2.4 (line 12). Then, during the second phase (lines 13–18), HRZ considers the remaining affected objects, i.e., not in R. If the current time is more than w timestamps before the time horizon $o.t_h$ of an object o, HRZ essentially ignores o (line 14). For each other affected object, its processing (handled by HRZ_ProcessObject at line 15) differs significantly from XTR. Once it concludes, HRZ checks whether the object should be included in the result set R provided that its trajectory distance has sufficiently decreased (lines 16–18).

We next describe the HRZ_ProcessObject procedure, shown in Algorithm 13. As in XTR, the procedure removes expired location distances if an event from Q was triggered (lines 1–2). The main operations of the procedure occur when either an object or a query location update were received (lines 3–23). First, the object's location is updated, if it changed, and its location distance is computed (lines 4–5). If the object o under processing did not belong in the result at the previous timestamp (line 6), the procedure computes the time horizon t_h by applying Lemma 5.2.5 (line 7); otherwise t_h is set to current time (line 8), meaning that object o may belong in the result. Since the time horizon is now recalculated taking into account the object's current location distance, it is necessary to check again if the object should be ignored (line 9). If the check succeeds, all stored information for object o is

cleared, its entry in the event queue is removed and an infinite trajectory distance is returned (lines 10–12).

ALGORITHM 13: HRZ_ProcessObject

```
1  if OExp event for o was triggered then
2  |  remove the expired location distance from o.hist
3  end
4  if QUpd or OUpd event for o was received then
5  |  update o.loc, if changed
6  |  compute new location distance d
7  |  if o ∉ R then
8  |  |  compute t_h
9  |  else t_h ← t
10 |  if t < t_h − w then
11 |  |  clear state of o
12 |  |  remove o's entry in Q
13 |  |  return D_o ← ∞
14 |  else
15 |  |  add d to o.hist
16 |  |  remove from o.hist all location distances less than d and with
   |  |  timestamps before t − w
17 |  |  if t < t_h then
18 |  |  |  return D_o ← ∞
19 |  |  else
20 |  |  |  t' ← the earliest timestamp in o.hist
21 |  |  |  if Q contains OExp event for o then
22 |  |  |  |  update OExp's time to t' + w
23 |  |  |  else
24 |  |  |  |  insert in Q the event < o, t' + w >
25 |  |  |  end
26 |  |  end
27 |  end
28 end
29 return D_o ← earliest location distance in o.hist
```

In the following operations (lines 14–23), it holds that the current time is $t \geq t_h - w$, hence HRZ needs to store locations and location distances. The object's current location distance d is stored (line 14), and all location distances less than d are removed (line 15) as in XTR. Subsequently, if the current time falls in the interval $[t_h - w, t_h]$ (line 16), finding the actual trajectory distance during this interval is not necessary, as the object is guaranteed to not be in the result set. Therefore, HRZ simply returns an infinite trajectory distance (line 17) and, to increase efficiency, it does not create a corresponding expiration event. A consequence is that at future timestamps after the current time horizon, there may exist expired location distances. Therefore, the procedure may also have to remove such distances (line 15). Otherwise, if the current time is not before the time horizon (line 18), the processing is identical to XTR. That is, the earliest timestamp is identified, and the event queue is properly updated (lines 19–22). The last operation is to compute the

trajectory distance from the earliest location distance and return it (line 24). As a final note, observe that if the object was not in the result at the previous timestamp, its processing is identical to XTR, as its time horizon is set to current time (line 8).

5.2.4 Experimentation

To evaluate the efficiency of the proposed algorithms for the continuous nearest trajectories query, we conduct experiments using three real-world trajectory datasets. In the following, we first present the datasets used for the evaluation and then we report the results of our experiments.

5.2.4.1 Datasets

To cover a variety of cases regarding the shapes of trajectories, the type of the objects, and the speed and type of movement, we use three different real-world datasets in our experiments. We refer to these datasets as *Beijing taxis*, *Aegean ships*, and *Athens vehicles*. These datasets vary in their characteristics, ensuring that our methods are robust across diverse settings. For example, in the *Beijing taxis* dataset, the shape of the trajectories exhibits a relatively high regularity due to the grid-like structure of the underlying road network. At the other end, the Athens road network is highly irregular, resulting in diverse trajectories with constantly varying headings. Finally, the *Aegean ships* trajectory dataset comprises relatively long trajectories with medium degree of heading variations.

A typical issue in trajectory datasets is the often high variation of the sampling rate, caused, for example, by weak GPS signal, or when the user manually switches off their personal tracking devices (e.g., to save battery or for privacy). In our datasets, to reduce such gaps, when the time interval between two consecutive reported locations exceeds a specified threshold (set to 30 seconds) but is not greater than a maximum threshold (set to 120 seconds), we use linear interpolation to create intermediate location updates.

We next detail the used datasets.

- *Beijing taxis.* These trajectories are from the T-Drive trajectory dataset, which contains GPS tracking data from taxis moving in the area of Beijing [YZXS11, YZZ+10]. A total of 1,023,924 trajectories are used. These trajectories belong to a total of 569 taxis recorded in the period 2/2/2008 – 4/2/2008. Each trajectory comprises on average 3,017 points (i.e. location updates).

- *Aegean ships.* This dataset contains GPS tracking data from ships moving in the Aegean sea[5]. A total of 986,275 trajectories are used, obtained from 887 ships in the period 31/12/2008 – 02/01/2009. On average, each trajectory comprises 1,101 points.

- *Athens vehicles.* This dataset contains GPS tracking data from vehicles moving in the area of Athens, recorded in the context of the SimpleFleet project[6]. 667,421 trajectories are used, coming from 2,497 vehicles on 01/10/2012. Each trajectory comprises 157 points on average.

[5] http://www.chorochronos.org/?q=node/8
[6] http://www.simplefleet.eu/

5.2.4.2 Results

The goal of the experimental evaluation is to study the efficiency of the proposed algorithms, and in particular to compare the speedup achieved by the XTR and HRZ algorithms with respect to the more generic baseline BSL algorithm. For this purpose, we conduct a series of experiments, using the datasets previously described. The trajectory distance metric used in all experiments is the maximum of all valid location distances. We note that the performance of BSL is identical for all metrics, as the method is distance agnostic. On the other hand, XTR and HRZ have roughly the same performance for any extremum-defined trajectory distance metrics.

The main performance metric is the total execution time, i.e., the time spent processing a CNT query over its entire lifespan. To better investigate the performance gains of XTR and HRZ with respect to BSL, we also report their relative improvement in execution time, and the percentage of events (location updates and expirations) that they process compared to BSL. The investigated parameters affecting the performance of the algorithms is the number k of nearest trajectories requested, the size w of the time window, and the number $|O|$ of objects. In all settings, the reported performance metrics (time and number of events) are the average of 10 executions involving randomly selected query objects. The answer to a CNT query is calculated at each timestamp that an update or an expiration event occurs.

5.2.4.3 Varying the number of nearest trajectories

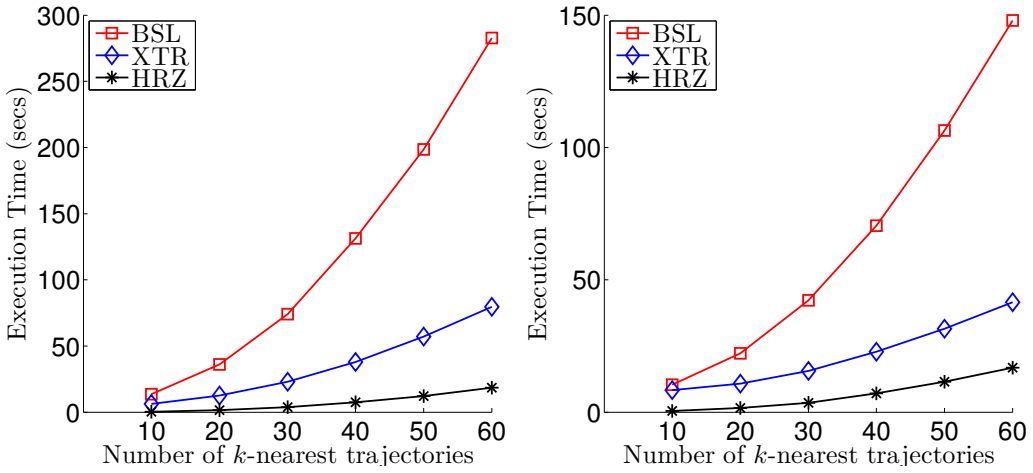
In this experiment, we measure the total execution time of each of the three algorithms with respect to the number k of nearest trajectories returned. The total monitoring time $\mathcal{T}$ is set to 60 minutes, and the size w of the time window for keeping each object's history is set to 5 minutes. The results are presented in Figure 5.6.

The first important observation is that for all datasets, the execution times of both XTR and HRZ are significantly lower than for BSL, clearly showing in practice the effectiveness of the corresponding optimizations for these cases. Furthermore, HRZ has also a clear benefit over XTR. The differences are more pronounced in the *Beijing taxis* dataset, which shows that, due to the regularity in the movement of objects imposed by the underlying grid-like structure of the Beijing road network, more effective pruning of location updates and distance recomputations can be achieved. In contrast, the differences become relatively smaller in *Athens vehicles*, where the road network is less uniform.

A second observation is that for all algorithms the execution time increases with k. This is expected since k regulates the size of the ordered list R that has to be maintained by the algorithm at each timestamp. However, this increase is lower for XTR and, even more so for HRZ, which is an additional evidence that XTR and HRZ need to process fewer events, and hence perform fewer lookup and sort operations on R.

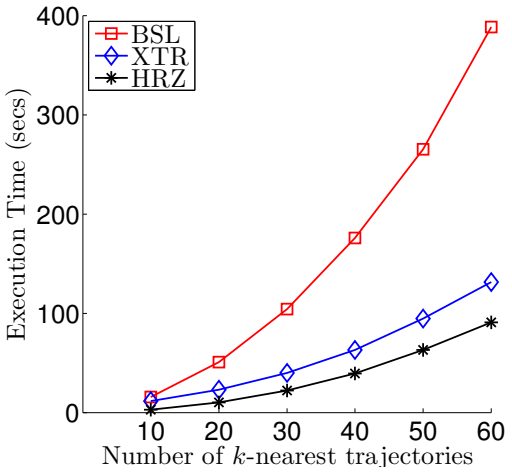
5.2.4.4 Varying the size of the time window

In the next experiment, we compare the execution time of the three algorithms with respect to the window size w during which the past location distances of an object remain valid and contribute to the trajectory distance. As previously, the total



(a) Beijing taxis

(b) Aegean ships



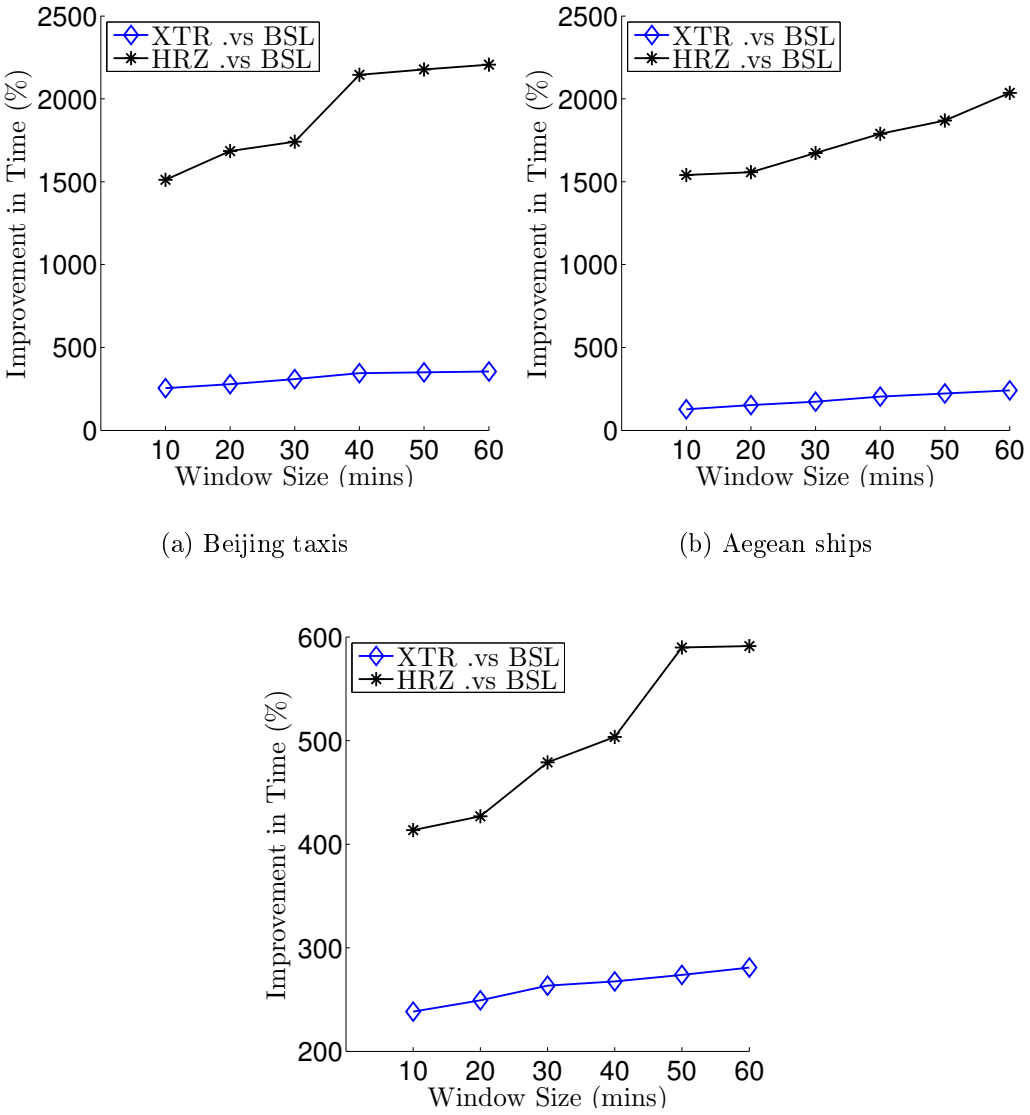
(c) Athens vehicles

Figure 5.6: *Execution time of BSL, XTR and HRZ w.r.t. the number k of nearest trajectories returned at each timestamp.*

monitoring time $\mathcal{T}$ was set to 60 minutes, and k was set to 10. To better illustrate the improvement in execution time achieved by XTR and HRZ with respect to BSL, we plot the speedup of XTR and HRZ compared to BSL. The results are shown in Figure 5.7.

As illustrated, XTR shows a speedup of almost up to 5 times over BSL, while for HRZ it is even higher, in the range of 15× to 22× for the first two datasets and 4× to 6× for the *Athens vehicles*. Notice that in this setting $k = 10$, so when these results are considered in conjunction with those illustrated in Figure 5.6, these speedups are expected to be increasingly higher for higher values of k.

Moreover, the speedup for both algorithms increases as the window size w increases. This behavior is because XTR and HRZ only consider the maximum or minimum value in each object's history, so the gain is higher for larger time windows. The gain for HRZ is even higher as w increases, since HRZ is able to set time horizons for objects later in the future, thus ignoring more location updates and



(a) Beijing taxis

(b) Aegean ships

(c) Athens vehicles

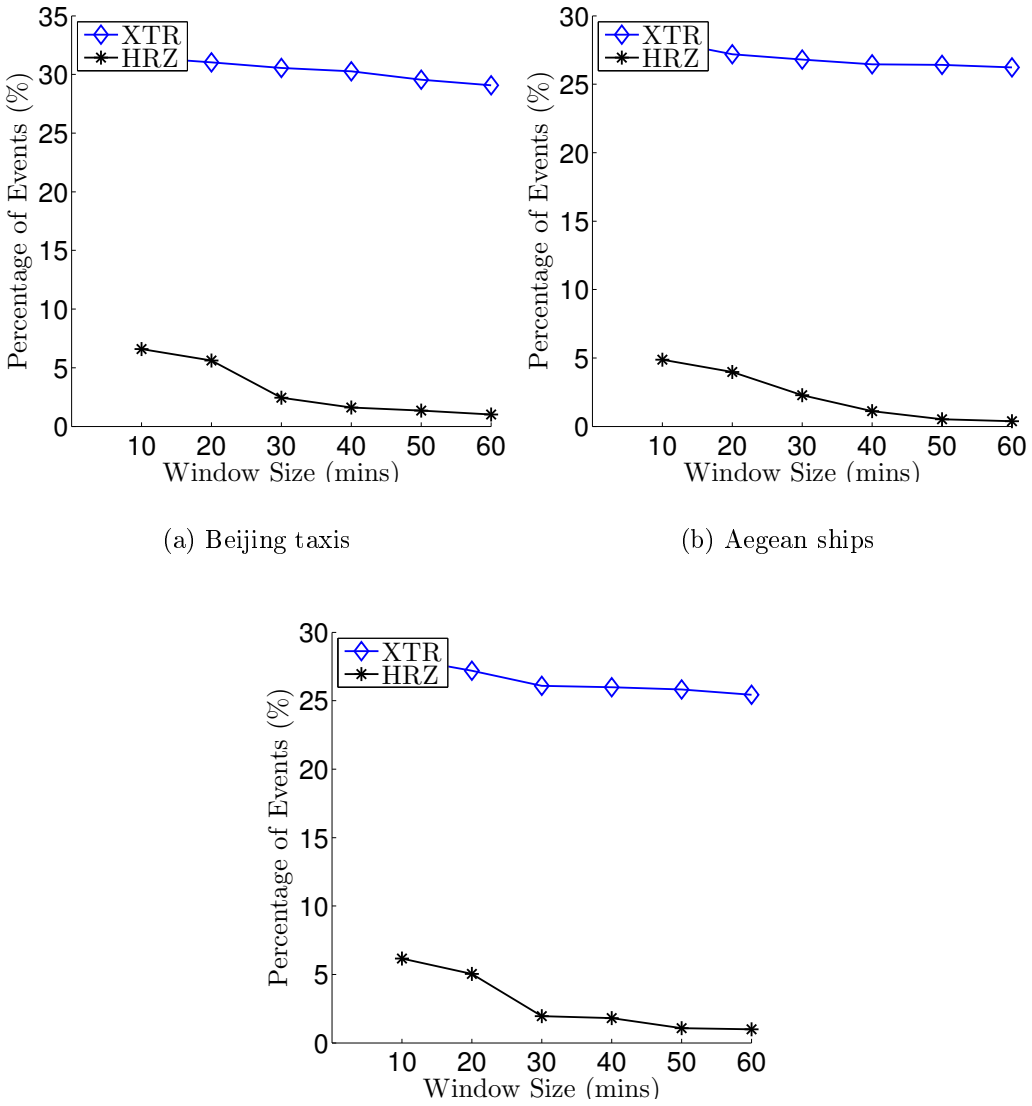
Figure 5.7: *Execution time speedup of XTR and HRZ compared to BSL w.r.t. the size w of the time window.*

further decreasing the total events to be handled.

To better illustrate the reduction of the number of events that XTR and HRZ process, and how this is affected by the size of the time window, we also report the number of events in the event queue Q that are created and processed by XTR and HRZ with respect to those by BSL. The results are shown in Figure 5.8. Indeed, the results are in agreement with those in Figure 5.7, showing that XTR needs to process only about 30% of the events processed by BSL, while HRZ fewer than 5%.

5.2.4.5 Varying the number of objects

In the last set of experiments, we measure the performance of the algorithms with respect to the number of objects. For this purpose, we create subsets of the original datasets, containing a specific portion of randomly selected objects, and ran the



(a) Beijing taxis

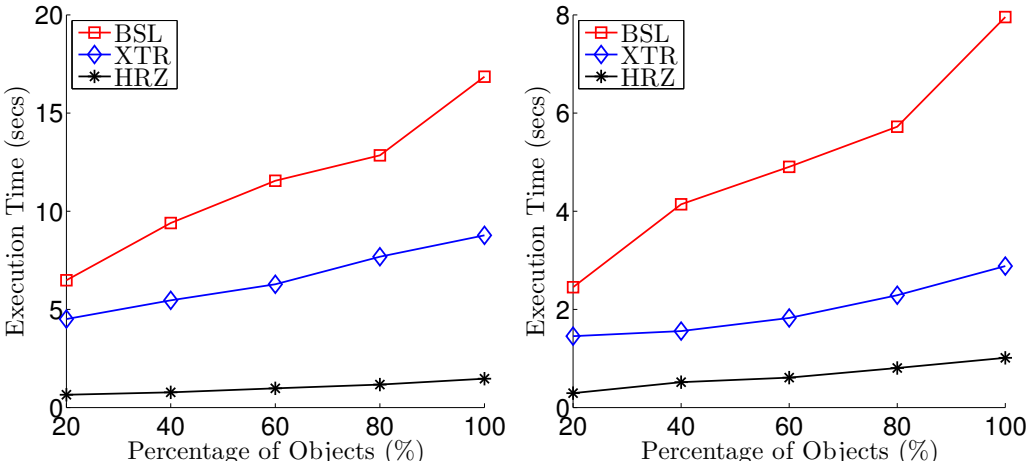
(b) Aegean ships

(c) Athens vehicles

Figure 5.8: *Percentage of events handled by XTR and HRZ compared to BSL w.r.t. the size w of the time window.*

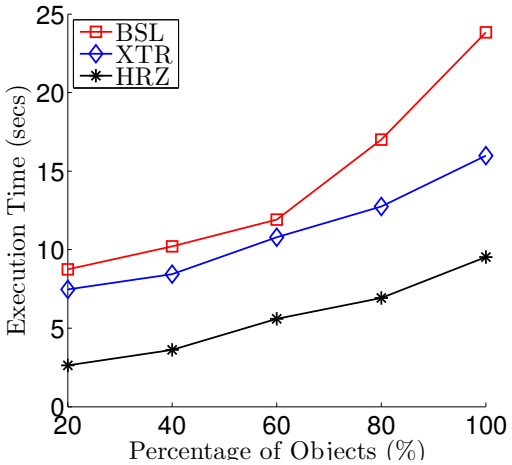
algorithms on these subsets. The other parameters are set to $\mathcal{T} = 60$ minutes, $k = 10$, and $w = 5$ minutes. The results are plotted in Figure 5.9.

As expected, the execution time of all algorithms increases as the size of the dataset increases. However, XTR and, especially, HRZ show better scalability. Especially HRZ for the cases of the *Beijing taxis* and the *Aegean ships*, where the movement of the objects is relatively more regular, proves to be quite robust with respect to the total number of objects, which verifies that it can successfully avoid unnecessary examinations of objects that cannot qualify as candidates for the result set.



(a) Beijing taxis

(b) Aegean ships



(c) Athens vehicles

Figure 5.9: *Execution time of BSL, XTR and HRZ w.r.t. the size $|O|$ of the dataset.*

Chapter 6

Conclusions and Future Directions

In this dissertation we have covered a lot of ground. We presented our work on mining and modeling geospatial knowledge from user generated content in the forms of textual data. We employed the extracted and modeled VGI in order to solve two significant research problems, namely, the location estimation of unknown POIs problem, and the popular path computation problem. Finally, we provided several competitive algorithms which solve the KNN trajectory problem for both non-continuous and continuous query cases. In the remainder of this chapter, we will further briefly summarize our contributions and we will identify potential extensions of our results for future work.

6.1 Mining and Modeling Geospatial Data

The increase in available user-generated data provides a unique opportunity for the generation of rich datasets in geographical information science. In this dissertation, we provide and approach for the extraction of geospatial knowledge from user generated texts, and a quantitative approach for the representation of qualitative spatial relations extracted from such data based on training probabilistic models. The proposed scheme returns estimates of uncertain object locations based on distance and orientation features as provided by human reporters in relation to known object locations. To achieve these desiderata, we propose a greedy learning algorithm based on the Expectation Maximization (EM) framework to train probabilistic models over spatial relationships; here, we restrict our attention on GMM models. The proposed approach seems to be promising in terms of accurately capturing and representing spatial relationships. Distance and orientation features tend to describe all spatial relations that were extracted from user generated texts in an informative way. Moreover, our probabilistic approach seems to be robust in handling any uncertainties, which characterize observations in crowd-sourced text data. As a future research direction, we already have been investigating new NLP techniques for the optimization of automatic extraction of POIs and spatial relationship information from texts. Moreover, we already investigate deep learning and advanced machine learning methods in order to handle the inherent uncertainty in user generated content.

6.2 Location Estimation

In this dissertation, our specific contribution is detecting spatial relationships in textual narratives and using them to "triangulate" the position of unknown object locations my employing probabilistic models and fusion algorithms. This is a first step for solving the emerging geocoding problem on the Internet. We introduced specific techniques for extracting spatial relations from textual narratives and use a novel quantitative approach based on training probabilistic models for the representation of spatial relations. Combining these models and interpreting them as observations in a location fusion algorithm allows us to reason about unknown object locations. The results show that "colloquial" location estimation facilitated by crowdsourced geospatial narratives is a feasible approach.

Directions for future work include the optimization of the location fusion algorithms in the direction of deep learning. Furthermore we will investigate the implementation of global prediction models, which could complement geocoding methods in our increasingly non-cartesian world. Also, this will enable us to evaluate additional probabilistic and deterministic modeling techniques and to develop more efficient text-to-map applications.

6.3 Popular Path Computation

In this dissertation, we presented new approaches to computing knowledge-enriched paths within road networks. We incorporated novel methods to extract spatial relations between pairs of Points of Interest such as "near" or "close by" from crowdsourced textual data, namely travel blogs. We quantified the extracted relations using probabilistic models to handle the inherent uncertainty of user-generated content. Based on these models, we proposed a new cost function to enrich real world road networks, based on Djikstra and skyline path computation. The new cost function reflects the closeness aspect according to the crowd. In contrast to existing approaches, we did not enrich previously computed paths with semantical information, but the entire network. Continuingly, two routing algorithms were presented taking this closeness aspect into account. Finally, we evaluated our ideas on two real world road network datasets, i.e., Paris, France, and New York City, USA. We used metadata from geotagged Flickr photos as a ground truth to support our initial goal of providing more popular paths. All our approaches performed very well by providing slightly longer paths but with significantly higher values of popularity.

For future work, we are researching alternative methods for aggregating all categories of spatial relations. Furthermore, we would like to investigate ways to suggest the popular path descriptions to the user based on the Points of Interest they will encounter underway.

6.4 Mining GPS Data

6.4.1 Non-Continuous KNN Queries

Efficient spatio-temporal data analysis is crucial for exploiting the massive amounts of spatio-temporal data that are becoming available in modern applications. In this

dissertation, we have addressed the problem of efficiently identifying the k-nearest neighbors of moving objects, taking into consideration proximity, direction and time. Starting with an exact algorithm, we have investigated two approximate algorithms which allow to faster identify an approximate set of k-NNs. The first employs a line simplification step to reduce the number of line segments to be examined when comparing trajectories. The second accelerates the query execution time by estimating prior probabilities based on pre-computed probability density functions of the trajectory line segments. Our experimental evaluation, conducted on three real-world datasets, has shown that these algorithms provide a favorable trade-off between execution time and accuracy for various cases.

Our current and future work focuses on combining deterministic and probabilistic machine learning techniques to achieve additional improvements in terms of accuracy, as well as extending these techniques to address other related problems in mining moving object trajectories.

6.4.2 Continuous KNN Queries

In the last part of this dissertation, we introduced and studied the problem of continuously reporting moving objects with similar recent trajectories to a given query object. This problem extends the case of continuous nearest neighbor monitoring and of discovering similar trajectories in historical data. We proposed a generic baseline method that operates for any aggregate trajectory distance metric; the extension to other metrics is left as future work. Then we turned our attention to instances where the distance between the trajectories of two objects is determined by the extrema (minimum and maximum) of their individual location distances. For these instances, we described two more efficient algorithms, with the latter taking into account a given bound on the velocities of objects. Our experimental study on real-world datasets showed that our methods exhibit up to 22 times performance gain compared to the baseline.

Our current and future work focuses on combining our current algorithms with efficient index data structures in order to optimize in terms of search speed and accuracy.

Bibliography

[AB09] Alan Akbik and Jürgen Broß. Wanderlust: Extracting semantic relations from natural language text using dependency grammar patterns. In *Proc. of the Workshop on Semantic Search*, SemSearch, 2009.

[ABK+07] Luis Otavio Alvares, Vania Bogorny, Bart Kuijpers, Jose Antonio Fernandes de Macedo, Bart Moelans, and Alejandro Vaisman. A model for enriching trajectories with semantic geographical information. In *Proc. of the 15th Annual ACM Int'l Symp. on Advances in Geographic Information Systems*, pages 22:1–22:8, 2007.

[AGLW08] Mattias Andersson, Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. Reporting leaders and followers among trajectories of moving point objects. *GeoInformatica*, 12(4):497–528, 2008.

[AJTY13] Ove Andersen, Christian S. Jensen, Kristian Torp, and Bin Yang. Ecotour: Reducing the environmental footprint of vehicles using eco-routes. In *MDM13*, pages 338–340, 2013.

[And13] Andrienko N. Fuchs G. Olteanu Raimond A. M. Symanzik J. Ziemlicki C. Andrienko, G. Extracting semantics of individual places from movement data by analyzing temporal patterns of visits. In *Proc. of the 1st ACM Int'l Workshop on Computational Models of Place (COMP) '13*, COMP '13, pages 9:9–9:16, New York, NY, USA, 2013. ACM.

[BC96] Donald J. Berndt and James Clifford. Finding patterns in time series: A dynamic programming approach. In *Advances in Knowledge Discovery and Data Mining*, pages 229–248. 1996.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 2006.

[BJKx06] Rimantas Benetis, S. Jensen, Gytis Karĉiauskas, and Simonas Ŝaltenis. Nearest and reverse nearest neighbor queries for moving objects. *The VLDB Journal*, 15(3):229–249, September 2006.

[BM06] Razvan Bunescu and Raymond J. Mooney. Subsequence kernels for relation extraction. In *Advances in Neural Information Processing Systems*, pages 171–178, 2006.

[BPPP08] Gloria Bordogna, Marco Pagani, Gabriella Pasi, and Giuseppe Psaila. Evaluating uncertain location-based spatial queries. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, pages 1095–1100, New York, NY, USA, 2008. ACM.

[BT08] Petko Bakalov and Vassilis J. Tsotras. Geosensor networks. chapter Continuous Spatiotemporal Trajectory Joins, pages 109–128. Springer-Verlag, Berlin, Heidelberg, 2008.

[BW01] Shivnath Babu and Jennifer Widom. Continuous queries over data streams. *SIGMOD Rec.*, 30(3):109–120, September 2001.

[CCL10] Zhiyuan Cheng, James Caverlee, and Kyumin Lee. You are where you tweet: A content-based approach to geo-locating twitter users. In *Proc. of the 19th ACM Int'l Conf. on Information and Knowledge Management*, CIKM, pages 759–768, 2010.

[CLE+13] Jaeyoung Choi, Howard Lei, Venkatesan Ekambaram, Pascal Kelm, Luke Gottlieb, Thomas Sikora, Kannan Ramchandran, and Gerald Friedland. Human vs machine: Establishing a human baseline for multimodal location estimation. In *Proc. of the 21st ACM Int'l Conf. on Multimedia*, MM, pages 867–876, 2013.

[CLEL12] Hau Wen Chang, Dongwon Lee, M. Eltaher, and Jeongkyu Lee. @phillies tweeting from philly? predicting twitter user locations with spatial word usage. In *IEEE/ACM Int'l Conf. on Advances in Social Networks Analysis and Mining*, ASONAM, pages 111–118, 2012.

[CMC07] Huiping Cao, Nikos Mamoulis, and David W. Cheung. Discovery of periodic patterns in spatiotemporal sequences. *IEEE Trans. Knowl. Data Eng.*, 19(4):453–467, 2007.

[CÖO05a] Lei Chen, M. Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD Conference*, pages 491–502, 2005.

[COO05b] Lei Chen, M. Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 491–502, New York, NY, USA, 2005. ACM.

[CSZ11] Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou. Discovering popular routes from trajectories. In *ICDE11*, pages 900–911, April 2011.

[CZLZ12] Muhammad Aamir Cheema, Wenjie Zhang, Xuemin Lin, and Ying Zhang. Efficiently processing snapshot and continuous reverse k nearest neighbors queries. *VLDB J.*, 21(5):703–728, 2012.

[DHS01] R. Duda, P. Hart, and D. Stork. *Pattern Classification.* 2001.

[DK] Matt Duckham and Lars Kulik. Simplest paths: Automated route selection for navigation. In *COSIT03*, pages 169–185.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[DP10] Euthymios Drymonas and Dieter Pfoser. Geospatial route extraction from texts. In *Proc. of the 1st Int'l Workshop on Data Mining for Geoinformatics*, DMGI, pages 29–37, 2010.

[Ege89] MaxJ. Egenhofer. A formal definition of binary topological relationships. In *Foundations of Data Organization and Algorithms*, volume 367 of *Lecture Notes in Computer Science*, pages 457–472. 1989.

[EH90] M. J. Egenhofer and J. Herring. A mathematical framework for the definitions of topological relationships. In *Int'l Symp. on Spatial Data Handling*, 1990.

[ES93] MaxJ. Egenhofer and Jayant Sharma. Topological relations between regions in r^2 and z^2. In *Advances in Spatial Databases*, volume 692 of *Lecture Notes in Computer Science*, pages 316–336. 1993.

[FGG+99] Andrew Frank, Stephane Grumbach, Ralf Hartmut Güting, Christian S. Jensen, Manolis Koubarakis, Nikos Lorentzos, Yannis Manolopoulos, Enrico Nardelli, Barbara Pernici, Hans-Jörg Schek, Michel Scholl, Timos Sellis, Babis Theodoulidis, and Peter Widmayer. Chorochronos: A research network for spatiotemporal database systems. *SIGMOD Rec.*, 28(3):12–21, September 1999.

[FGPT07a] Elias Frentzos, Kostas Gratsias, Nikos Pelekis, and Yannis Theodoridis. Algorithms for nearest neighbor search on moving object trajectories. *GeoInformatica*, 11(2):159–193, 2007.

[FGPT07b] Elias Frentzos, Kostas Gratsias, Nikos Pelekis, and Yannis Theodoridis. Algorithms for nearest neighbor search on moving object trajectories. *Geoinformatica*, 11(2):159–193, June 2007.

[FGT07] Elias Frentzos, Kostas Gratsias, and Yannis Theodoridis. Index-based most similar trajectory search. In *ICDE*, pages 816–825, 2007.

[FKS03] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top k lists. *SIAM J. Discrete Math.*, 17(1):134–160, 2003.

[FLN03] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):614–656, 2003.

[FSE11] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, EMNLP, pages 1535–1545, 2011.

[FSSR13] Dan Feldman, Andrew Sugaya, Cynthia Sung, and Daniela Rus. idiary: From gps signals to a text-searchable diary. In *Proc. of the*

11th ACM Conf. on Embedded Networked Sensor Systems, pages 6:1–6:12, 2013.

[FVD10] Gerald Friedland, Oriol Vinyals, and Trevor Darrell. Multimodal location estimation. In *Proc. of the 18th ACM Int'l Conf. on Multimedia*, MM, pages 1245–1252, 2010.

[G̈94] Ralf Hartmut Güting. An introduction to spatial database systems. *The VLDB Journal*, 3(4):357–399, 1994.

[GAL+10] Ander Garcia, Olatz Arbelaitz, Maria Teresa Linaza, Pieter Vansteenwegen, and Wouter Souffriau. *Personalized tourist route generation.* Springer, 2010.

[GBX10a] Ralf Hartmut Güting, Thomas Behr, and Jianqiu Xu. Efficient k-nearest neighbor search on moving object trajectories. *VLDB J.*, 19(5):687–714, 2010.

[GBX10b] Ralf Hartmut Güting, Thomas Behr, and Jianqiu Xu. Efficient k-nearest neighbor search on moving object trajectories. *The VLDB Journal*, 19(5):687–714, October 2010.

[GKMP14] Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou. A survey on algorithmic approaches for solving tourist trip design problems. *Journal of Heuristics 20*, (3):291–328, 2014.

[GKRS] Franz Graf, Hans-Peter Kriegel, Matthias Renz, and Matthias Schubert. Mario: Multi-attribute routing in open street map. In *SSTD'11*, pages 486–490.

[GLC+06] Yunjun Gao, Chun Li, Gencai Chen, Ling Chen, Xianta Jiang, and Chun Chen. Bfpknn: An efficient k-nearest-neighbor search algorithm for historical moving object trajectories. In *Proceedings of the 4th International Conference on Advances in Information Systems*, ADVIS'06, pages 70–79, Berlin, Heidelberg, 2006. Springer-Verlag.

[GLC+07a] Yunjun Gao, Chun Li, Gencai Chen, Ling Chen, Xianta Jiang, and Chun Chen. Efficient k-nearest-neighbor search algorithms for historical moving object trajectories. *J. Comput. Sci. Technol.*, 22(2):232–244, 2007.

[GLC+07b] Yunjun Gao, Chun Li, Gencai Chen, Qing Li, and Chun Chen. Efficient algorithms for historical continuous k nn query processing over moving object trajectories. In *APWeb/WAIM*, pages 188–199, 2007.

[GLW08] Joachim Gudmundsson, Patrick Laube, and Thomas Wolle. Movement patterns in spatio-temporal data. In *Encyclopedia of GIS*, pages 726–732. 2008.

[GvK06a] Joachim Gudmundsson and Marc van Kreveld. Computing longest duration flocks in trajectory data. In *Proceedings of the 14th Annual*

ACM International Symposium on Advances in Geographic Information Systems, GIS '06, pages 35–42, New York, NY, USA, 2006. ACM.

[GvK06b] Joachim Gudmundsson and Marc J. van Kreveld. Computing longest duration flocks in trajectory data. In *GIS*, pages 35–42, 2006.

[GZC+11] Yunjun Gao, Baihua Zheng, Gang Chen, Chun Chen, and Qing Li. Continuous nearest-neighbor search in the presence of obstacles. *ACM Trans. Database Syst.*, 36(2):9:1–9:43, June 2011.

[HCB12] Bo Han, Paul Cook, and Timothy Baldwin. Geolocation prediction in social media data by finding location indicative words. In *Proc. of 24th Int'l Conf. on Computational Linguistics*, COLING, pages 1045–1062, 2012.

[HCB14] Bo Han, Paul Cook, and Timothy Baldwin. Text-based twitter user geolocation prediction. *Journal of Artificial Intelligence Research*, pages 451–500, 2014.

[HE08] James Hays and Alexei A. Efros. im2gps: estimating geographic information from a single image. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, CVPR, 2008.

[Hir97] Daniel S. Hirschberg. Pattern matching algorithms. chapter Serial Computations of Levenshtein Distances, pages 123–141. 1997.

[HLL08] Yuan-Ko Huang, Shi-Jei Liao, and Chiang Lee. Efficient continuous k-nearest neighbor query processing over moving objects with uncertain speed and direction. In *Proceedings of the 20th International Conference on Scientific and Statistical Database Management*, SSDBM '08, pages 549–557, Berlin, Heidelberg, 2008. Springer-Verlag.

[HLL09] Yuan-Ko Huang, Shi-Jei Liao, and Chiang Lee. Evaluating continuous k-nearest neighbor query on moving objects with uncertainty. *Inf. Syst.*, 34(4-5):415–437, 2009.

[ISS03] Glenn S. Iwerks, Hanan Samet, and Ken Smith. Continuous k-nearest neighbor queries for continuously moving points with updates. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, pages 512–523. VLDB Endowment, 2003.

[JFS+15] Gregor Jossé, Maximilian Franzke, Georgios Skoumas, Andreas Züfle, Mario A. Nascimento, and Matthias Renz. A framework for computation of popular paths from crowdsourced data. In *ICDE15*, pages 1428–1431, 2015.

[JPS13] A. Jaiswal, Wei Peng, and Tong Sun. Predicting time-sensitive user locations from social media. In *IEEE/ACM Int'l Conf. on Advances in Social Networks Analysis and Mining*, ASONAM, pages 870–877, 2013.

[JSZ08] Hoyoung Jeung, Heng Tao Shen, and Xiaofang Zhou. Convoy queries in spatio-temporal databases. In *ICDE*, pages 1457–1459, 2008.

[JYZ+08a] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S. Jensen, and Heng Tao Shen. Discovery of convoys in trajectory databases. *PVLDB*, 1(1):1068–1080, 2008.

[JYZ+08b] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, Christian S. Jensen, and Heng Tao Shen. Discovery of convoys in trajectory databases. *Proc. VLDB Endow.*, 1(1):1068–1080, August 2008.

[KEG93] Wolfgang Kainz, Max J. Egenhofer, and Ian Greasley. Modeling spatial relations and operations with partially ordered sets. *Int'l Journal of Geographical Information Systems*, 7:215–229, 1993.

[KG90] Maurice Kendall and Jean D. Gibbons. *Rank Correlation Methods.* A Charles Griffin Title, 5 edition, September 1990.

[KGT99] George Kollios, Dimitrios Gunopulos, and Vassilis J. Tsotras. Nearest neighbor queries in a mobile environment. In *Proceedings of the International Workshop on Spatio-Temporal Database Management*, STDBM '99, pages 119–134, London, UK, UK, 1999. Springer-Verlag.

[KL51] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 1951.

[KMB05a] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. On discovering moving clusters in spatio-temporal data. In *SSTD*, pages 364–381, 2005.

[KMB05b] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. On discovering moving clusters in spatio-temporal data. In *Proceedings of the 9th International Conference on Advances in Spatial and Temporal Databases*, SSTD'05, pages 364–381, Berlin, Heidelberg, 2005. Springer-Verlag.

[KMM+06] Dmitri V. Kalashnikov, Yiming Ma, Sharad Mehrotra, Ramaswamy Hariharan, and Carter Butts. Modeling and querying uncertain spatial information for situational awareness applications. In *Proc. of the 14th annual ACM Int'l Symposium on Advances in Geographic Information Systems*, GIS, pages 131–138, 2006.

[KRS] Hans-Peter Kriegel, Matthias Renz, and Matthias Schubert. Route skyline queries: a multi-preference path planning approach. In *ICDE10*, pages 261–272.

[KSC+13] Pascal Kelm, Sebastian Schmiedeke, Jaeyoung Choi, Gerald Friedland, Venkatesan Nallampatti Ekambaram, Kannan Ramchandran, and Thomas Sikora. A novel fusion method for integrating multiple modalities and knowledge for multimodal location estimation. In *Proc. of the 2nd ACM Int'l Workshop on Geotagging and Its Applications in Multimedia*, GeoMM, pages 7–12, 2013.

[KSF+03] Manolis Koubarakis, Timos K. Sellis, Andrew U. Frank, Stéphane Grumbach, Ralf Hartmut Güting, Christian S. Jensen, Nikos A. Lorentzos, Yannis Manolopoulos, Enrico Nardelli, Barbara Pernici, Hans-Jörg Schek, Michel Scholl, Babis Theodoulidis, and Nectaria Tryfona, editors. *Spatio-Temporal Databases: The Chorochronos Approach*, Lecture Notes in Computer Science, 2003.

[KVOM11] Parisa Kordjamshidi, Martijn Van Otterlo, and Marie-Francine Moens. Spatial role labeling: Towards extraction of spatial relations from natural language. *ACM Trans. Speech Lang. Process.*, 8(3):4:1–4:36, 2011.

[LB99] Jonathan Q. Li and Andrew R. Barron. Mixture density estimation. In *Advances in Neural Information Processing Systems 12*, pages 279–285, 1999.

[LB02] Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proc. of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP, pages 63–70, 2002.

[LC09] Xiang Lian and Lei Chen. Efficient processing of probabilistic reverse nearest neighbor queries over uncertain data. *VLDB J.*, 18(3):787–808, 2009.

[LCC12] Mingqi Lv, Ling Chen, and Gencai Chen. Discovering personally semantic places from gps trajectories. In *Proc. of the 21st ACM Int'l Conf. on Information and Knowledge Management*, pages 1552–1556, 2012.

[LDH+10] Zhenhui Li, Bolin Ding, Jiawei Han, Roland Kays, and Peter Nye. Mining periodic behaviors for moving objects. In *KDD*, pages 1099–1108, 2010.

[LDHK10a] Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. Swarm: Mining relaxed temporal moving object clusters. *PVLDB*, 3(1):723–734, 2010.

[LDHK10b] Zhenhui Li, Bolin Ding, Jiawei Han, and Roland Kays. Swarm: Mining relaxed temporal moving object clusters. *Proc. VLDB Endow.*, 3(1-2):723–734, September 2010.

[LHL08] Jae-Gil Lee, Jiawei Han, and Xiaolei Li. Trajectory outlier detection: A partition-and-detect framework. In *ICDE*, pages 140–149, 2008.

[LHW07a] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *SIGMOD Conference*, pages 593–604, 2007.

[LHW07b] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: A partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 593–604, New York, NY, USA, 2007. ACM.

[LIR+12] Corrado Loglisci, Dino Ienco, Mathieu Roche, Maguelonne Teisseire, and Donato Malerba. An unsupervised framework for topological relations extraction from geographic documents. In *Database and Expert Systems Applications*, volume 7447 of *Lecture Notes in Computer Science*, pages 48–55. 2012.

[LS05] Bin Lin and Jianwen Su. Shapes based trajectory queries for moving objects. In *Proceedings of the 13th Annual ACM International Workshop on Geographic Information Systems*, GIS '05, pages 21–30, New York, NY, USA, 2005. ACM.

[LS08] Bin Lin and Jianwen Su. One Way Distance: For shape based similarity search of moving object trajectories. *GeoInformatica*, 12(2):117–142, 2008.

[MGM14] Ludovic Moncla, Mauro Gaio, and Sébastien Mustière. Automatic itinerary reconstruction from texts. In *Geographic Information Science*, volume 8728 of *Lecture Notes in Computer Science*, pages 253–267, 2014.

[MKM08] Yiming Ma, Dmitri V. Kalashnikov, and Sharad Mehrotra. Toward managing uncertain spatial information for situational awareness applications. *IEEE Trans. on Knowl. and Data Eng.*, 20(10):1408–1423, 2008.

[MPH05] Kyriakos Mouratidis, Dimitris Papadias, and Marios Hadjieleftheriou. Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 634–645, New York, NY, USA, 2005. ACM.

[MPTG09] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. WhereNext: a location predictor on trajectory pattern mining. In *KDD*, pages 637–646, 2009.

[MRANIG14] Ludovic Moncla, Walter Renteria-Agualimpia, Javier Nogueras-Iso, and Mauro Gaio. Geocoding for texts with fine-grain toponyms: An experiment on a geoparsed hiking description corpus. In *Proc. of the 22nd ACM Int'l Conf. on Advances in Geographic Information Systems*, SIGSPATIAL, pages 253–267, 2014.

[MSB13] Filipe Mesquita, Jordan Schmidek, and Denilson Barbosa. Effectiveness and efficiency of open relation extraction. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, EMNLP, pages 447–457, 2013.

[MSWHD] Hatem Mousselly-Sergieh, Daniel Watzinger, Bastian Huber, and Mario et. al Döller. World-wide scale geotagged image dataset for automatic image annotation and reverse geotagging. In *ACM MMSys14*, pages 47–52.

[NK09] Paul Newson and John Krumm. Hidden markov map matching through noise and sparseness. In *ACM SIGSPATIAL GIS 09*, pages 336–343, 2009.

[NZE+13] Johannes Niedermayer, Andreas Züfle, Tobias Emrich, Matthias Renz, Nikos Mamoulis, Lei Chen, and Hans-Peter Kriegel. Probabilistic nearest neighbor queries on uncertain moving object trajectories. *Proc. VLDB Endow.*, 7(3):205–216, November 2013.

[PBKA08] Andrey Tietbohl Palma, Vania Bogorny, Bart Kuijpers, and Luis Otavio Alvares. A clustering-based approach for discovering interesting places in trajectories. In *Proc. of the ACM Symp. on Applied Computing*, pages 863–868, 2008.

[PJT00a] Dieter Pfoser, Christian S. Jensen, and Yannis Theodoridis. Novel approaches in query processing for moving object trajectories. In *VLDB*, pages 395–406, 2000.

[PJT00b] Dieter Pfoser, Christian S. Jensen, and Yannis Theodoridis. Novel approaches in query processing for moving object trajectories. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 395–406, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[PKM+07a] Nikos Pelekis, Ioannis Kopanakis, Gerasimos Marketos, Irene Ntoutsi, Gennady Andrienko, and Yannis Theodoridis. Similarity search in trajectory databases. In *Proceedings of the 14th International Symposium on Temporal Representation and Reasoning*, TIME '07, pages 129–140, Washington, DC, USA, 2007. IEEE Computer Society.

[PKM+07b] Nikos Pelekis, Ioannis Kopanakis, Gerasimos Marketos, Irene Ntoutsi, Gennady L. Andrienko, and Yannis Theodoridis. Similarity search in trajectory databases. In *TIME*, pages 129–140, 2007.

[PS94] Dimitris Papadias and Timos Sellis. Qualitative representation of spatial knowledge in two-dimensional space. *The VLDB Journal*, 3(4):479–516, 1994.

[PSR+13] Christine Parent, Stefano Spaccapietra, Chiara Renso, Gennady Andrienko, Natalia Andrienko, Vania Bogorny, Maria Luisa Damiani, Aris Gkoulalas-Divanis, Jose Macedo, Nikos Pelekis, Yannis Theodoridis, and Zhixian Yan. Semantic trajectories modeling and analysis. *ACM Comput. Surv. '13*, 45(4):42:1–42:32, August 2013.

[PSTE95] Dimitris Papadias, Timos Sellis, Yannis Theodoridis, and Max J. Egenhofer. Topological relations in the world of minimum bounding rectangles: A study with r-trees. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, SIGMOD '95, pages 92–103, New York, NY, USA, 1995. ACM.

[PTS94] Dimitris Papadias, Yannis Theodoridis, and Timos Sellis. The retrieval of direction relations using r-trees. In *Database and Expert Systems Applications*, volume 856 of *Lecture Notes in Computer Science*, pages 173–182. 1994.

[QSA] Daniele Quercia, Rossano Schifanella, and Luca Maria Aiello. The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city. *CoRR14*, abs/1407.1031.

[RW14] Kai-Florian Richter and Stephan Winter. Cognitive aspects: How people perceive, memorize, think and talk about landmarks. In *Landmarks*, pages 41–108. Springer International Publishing, 2014.

[SB13] Dimitris Sacharidis and Panagiotis Bouros. Routing directions: Keeping it fast and simple. In *Proc. of the 21st ACM Int'l Conf. on Advances in Geographic Information Systems*, pages 164–173, 2013.

[SC06] W. Shi and C. K. Cheung. Performance evaluation of line simplification algorithms for vector generalization. *The Cartographic Journal*, 43(1):27–44, 2006.

[SJSK] Michael Shekelyan, Gregor Jossé, Matthias Schubert, and Hans-Peter Kriegel. Linear path skyline computation in bicriteria networks. In *DASFAA14*, pages 173–187.

[SJZ+15] Klaus Arthur Schmid, Gregor Jossé, Andreas Züfle, Georgios Skoumas, Matthias Schubert, and Dieter Pfoser. Tourismo: User preference driven touristic (trip) search engine. In *International Symposium on Spatial and Temporal Databases (SSTD)*, 2015.

[SP11] Stefano Spaccapietra and Christine Parent. Adding meaning to your steps. In *Proc. of the 30th Int'l Conf. on Conceptual Modeling*, pages 13–31, 2011.

[SPK13] Georgios Skoumas, Dieter Pfoser, and Anastasios Kyrillidis. On quantifying qualitative geospatial data: A probabilistic approach. In *Proc. of the 2nd ACM Int'l Workshop on Crowdsourced and Volunteered Geographic Information*, GEOCROWD, pages 71–78, 2013.

[SPKS15] Georgios Skoumas, Dieter Pfoser, Anastasios Kyrillidis, and Timos Sellis. Location estimation using crowdsourced spatial relations. In *Under submission*, pages 71–78, Submitted to ACM TSAS 2015.

[SR01] Zhexuan Song and Nick Roussopoulos. K-nearest neighbor search for moving query point. In *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*, SSTD '01, pages 79–96, London, UK, UK, 2001. Springer-Verlag.

[SSC90] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*, pages 167–193. 1990.

[SSJ+] Georgios Skoumas, Klaus Arthur Schmid, Gregor Jossé, Andreas Züfle, Mario A. Nascimento, Matthias Renz, and Dieter Pfoser. Towards knowledge-enriched path computation. In *ACM SIGSPATIAL GIS14 (to appear)*.

[SSJ+14] Georgios Skoumas, Klaus Arthur Schmid, Gregor Jossé, Andreas Züfle, Mario Nascimento, Matthias Renz, and Dieter Pfoser. Towards knowledge-enriched path computation. In *Proc. of the 22nd ACM Int'l Conf. on Advances in Geographic Information Systems*, 2014.

[SSJ+15] Georgios Skoumas, Klaus Arthur Schmid, Gregor Jossé, Mario Nascimento, Andreas Züfle, Matthias Renz, Dieter Pfoser, and Matthias Schubert. Knowledge-enriched route computation. In *International Symposium on Spatial and Temporal Databases (SSTD)*, 2015.

[SSS14] Dimitris Sacharidis, Dimitrios Skoutas, and Georgios Skoumas. Continuous monitoring of nearest trajectories. In *Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '14, pages 361–370, New York, NY, USA, 2014. ACM.

[SSV13] Georgios Skoumas, Dimitrios Skoutas, and Alexandra Vlachaki. Efficient identification and approximation of k-nearest moving neighbors. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 264–273, New York, NY, USA, 2013. ACM.

[TP02] Yufei Tao and Dimitris Papadias. Time-parameterized queries in spatio-temporal databases. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, SIGMOD '02, pages 334–345, New York, NY, USA, 2002. ACM.

[TTC+11] Goce Trajcevski, Roberto Tamassia, Isabel F. Cruz, Peter Scheuermann, David Hartglass, and Christopher Zamierowski. Ranking continuous nearest neighbors for uncertain trajectories. *VLDB J.*, 20(5):767–791, 2011.

[TTD+09] Goce Trajcevski, Roberto Tamassia, Hui Ding, Peter Scheuermann, and Isabel F. Cruz. Continuous probabilistic nearest-neighbor queries for uncertain trajectories. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09, pages 874–885, New York, NY, USA, 2009. ACM.

[VBT09a] Marcos R. Vieira, Petko Bakalov, and Vassilis J. Tsotras. On-line discovery of flock patterns in spatio-temporal data. In *GIS*, pages 286–295, 2009.

[VBT09b] Marcos R. Vieira, Petko Bakalov, and Vassilis J. Tsotras. On-line discovery of flock patterns in spatio-temporal data. In *Proceedings of*

the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS '09, pages 286–295, New York, NY, USA, 2009. ACM.

[VGK02a] Michail Vlachos, Dimitrios Gunopoulos, and George Kollios. Discovering similar multidimensional trajectories. In *Proceedings of the 18th International Conference on Data Engineering*, ICDE '02, pages 673–. IEEE Computer Society, 2002.

[VGK02b] Michail Vlachos, Dimitrios Gunopulos, and George Kollios. Discovering similar multidimensional trajectories. In *ICDE*, pages 673–684, 2002.

[VTS98] Michael Vazirgiannis, Yannis Theodoridis, and Timos Sellis. Spatio-temporal composition and indexing for large multimedia applications. *Multimedia Syst.*, 6(4):284–298, July 1998.

[VVK03] J. J. Verbeek, N. Vlassis, and B. Kröse. Efficient greedy learning of gaussian mixture models. *Neural Computation*, 15:469–485, 2003.

[WKB14] Jan Oliver Wallgrün, Alexander Klippel, and Timothy Baldwin. Building a corpus of spatial relational expressions extracted from web documents. In *Proceedings of the 8th Workshop on Geographic Information Retrieval*, GIR, pages 6:1–6:8, 2014.

[WLH06] Yida Wang, Ee-Peng Lim, and San-Yih Hwang. Efficient mining of group patterns from user movement data. *Data Knowl. Eng.*, 57(3):240–282, 2006.

[WM03] Yuhang Wang and Fillia Makedon. R-histogram: quantitative representation of spatial relations for similarity-based image retrieval. In *Proc. of the 11th ACM Int'l Conf. on Multimedia*, MULTIMEDIA, pages 323–326, 2003.

[WR11] Matthias Westphal and Jochen Renz. Evaluating and minimizing ambiguities in qualitative route instructions. In *Proc. of the 19th ACM Int'l Conf. on Advances in Geographic Information Systems*, pages 171–180, 2011.

[XMA05] Xiaopeng Xiong, Mohamed F. Mokbel, and Walid G. Aref. Sea-cnn: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, pages 643–654, Washington, DC, USA, 2005. IEEE Computer Society.

[YAS03a] Yutaka Yanagisawa, Jun-ichi Akahani, and Tetsuji Satoh. Shape-based similarity query for trajectory of mobile objects. In *Mobile Data Management*, pages 63–77, 2003.

[YAS03b] Yutaka Yanagisawa, Jun-ichi Akahani, and Tetsuji Satoh. Shape-based similarity query for trajectory of mobile objects. In *Proceedings of the 4th International Conference on Mobile Data Management*, MDM '03, pages 63–77, London, UK, UK, 2003. Springer-Verlag.

[YCP+11] Zhixian Yan, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, and Karl Aberer. Semitri: A framework for semantic annotation of heterogeneous trajectories. In *Proc. of the 14th Int'l Conf. on Extending Database Technology*, pages 259–270, 2011.

[YCP+13] Zhixian Yan, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, and Karl Aberer. Semantic trajectories: Mobility data computation and annotation. *ACM Trans. Intell. Syst. Technol.*, 4(3):49:1–49:38, July 2013.

[YPK05] Xiaohui Yu, Ken Q. Pu, and Nick Koudas. Monitoring k-nearest neighbor queries over moving objects. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, pages 631–642, Washington, DC, USA, 2005. IEEE Computer Society.

[YSC+10] Zhixian Yan, Lazar Spremic, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, and Karl Aberer. Automatic construction and multi-level visualization of semantic trajectories. In *Proc. of the 18th Int'l Conf. on Advances in Geographic Information Systems*, pages 524–525, 2010.

[Yua11] Yecheng Yuan. Extracting spatial relations from document for geographic information retrieval. In *Proc. of 19th Int'l Conf. on Geoinformatics*, pages 1–5, 2011.

[YZXS11] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. In *KDD*, pages 316–324, 2011.

[YZZ+10] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: driving directions based on taxi trajectories. In *GIS*, pages 99–108, 2010.

[ZAR02] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. In *Proc. of the ACL-02 Conf. on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP, pages 71–78, 2002.

[ZS97] Zhiyuan Zhao and Alan Saalfeld. Linear-time sleeve fitting polyline simplification algorithms. In *Autocarto 13, ACSM/ASPRS'97 Technical Papers*, pages 214–223, 1997.

[ZZDZ11] Xueying Zhang, Chunju Zhang, Chaoli Du, and Shaonan Zhu. Svm based extraction of spatial relations in text. In *Proc. of IEEE Int'l Conf. on Spatial Data Mining and Geographical Knowledge Services*, ICSDM, pages 529–533, 2011.