Multi-Source Spatial Entity Extraction and Linkage

Isaj, Suela

Publication date:
2021

Document Version
Publisher's PDF, also known as Version of record

MULTI-SOURCE SPATIAL ENTITY EXTRACTION AND LINKAGE

BY
SUELA ISAJ

DISSERTATION SUBMITTED 2021

AALBORG UNIVERSITY
DENMARK

Multi-Source Spatial Entity Extraction and Linkage

Ph.D. Dissertation
Suela Isaj

Dissertation submitted on March 10, 2021

A thesis submitted to the Technical Faculty of IT and Design at Aalborg University (AAU) and the Department of Computer and Decision Engineering at Université libre de Bruxelles (ULB), in partial fulfillment of the requirements within the scope of the IT4BI-DC programme for the joint Ph.D. degree in Computer Science. The thesis is not submitted to any other organization at the same time.

Dissertation submitted: March 10, 2021

AAU PhD Supervisor: Prof. Torben Bach Pedersen
Aalborg University, Denmark

ULB PhD Co-Supervisor: Prof. Esteban Zimányi
Université libre de Bruxelles

AAU PhD Committee: Associate Professor Manfred Jaeger (chair)
Aalborg University, Denmark

Associate Professor Konstantinos Stefanidis
University of Tampere, Finland

Associate Professor Maria Luisa Damiani
University of Milano, Italy

ULB PhD Committee: Assoc. Prof. Mahmoud Sakr
Université libre de Bruxelles

Assoc. Prof. Manfred Jaeger
Aalborg University, Denmark

Assoc. Prof. Konstantinos Stefanidis
University of Tampere, Finland

Assoc. Prof. Maria Luisa Damiani
University of Milano, Italy

PhD Series: Technical Faculty of IT and Design, Aalborg University
Department: Department of Computer Science

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-911-4

Published by:
Aalborg University Press
Kroghstræde 3
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Rosendahls, 2021

Abstract

Web data sources contain large amounts of geo-social data, consisting of users, friendship/follower networks, check-ins, reviews, locations, etc., which are of great interest to academia and industry. There are publicly available datasets of samples of this web data, but they are very old (over ten years), not large, and not rich in attributes. Alternatively, one could use the public APIs to access and download web data. Unfortunately, this process is challenging due to the APIs limitations (e.g. the amount of data retrieved in a request, the number of requests performed within a timeframe, etc.). Thus, there is a need for algorithms that, given the limitations, are able to retrieve a good quality dataset from web sources, a need that the current state-of-the-art does not address.

This thesis aims to provide algorithms and tools that can produce larger, recent, duplicate-free, and rich-in-attributes spatial entity data. To obtain larger and recent data from web data sources, we propose multi-source seed-driven (*MSSD*) algorithms that use the public free APIs to extract geo-social data. The *MSSD* algorithms aim to maximize the amount of data extracted while minimizing the number of requests and respecting each source's API limitations. The rationale behind the seed-driven algorithms is to perform some API requests for having an initial dataset and then use the points of the richest source as seed in the API requests for the rest of the sources. We propose different techniques for choosing the points and the radius of the search. We opt for a multi-source solution given that multiple sources provide independent information and diverse attributes as opposed to using only one source. Moreover, we experimentally demonstrate that using a single source algorithm sometimes converges to a dead end. The *MSSD* algorithms extract overall 14.3 times more data than the initial querying, and the optimized version *MSSD** retrieves 90% of the data with less than 16% of the requests of the non-optimized version.

When obtaining multi-source data, the same spatial entity might exist in different sources and sometimes even within the same source. These "duplicates" are not easy to detect since they have different attributes, they are expressed in different forms, and they might even contain contradicting at-

tribute values. The problem of finding which pairs of spatial entities refer to a real physical entity is referred to as *spatial entity linkage*. We address this problem with several algorithms, which all share the same spatial blocking technique, and they use skylines to rank the compared pairs. The spatial blocking technique (*QuadFlex*) that we propose is a quadtree-inspired algorithm that groups the spatial entities based on the distance between them and the area's density. Moreover, it allows the assignment of spatial entities in more than one child to not miss any relevant comparisons. The spatial entities that fall into the same child are compared pairwise.

To decide which pairs belong to the same physical entity, we propose novel skyline-based (SkyEx-*) algorithms, which use preference functions to assign skylines to the pairs. The threshold-based *SkyEx*, *SkyEx-F* and *SkyEx-FES* require a threshold that is the number of skylines to separate the positive from the negative class, and they are able to achieve an F-measure of 0.72 on the whole dataset and 0.85 on a manually labeled sample. We introduce a fully unsupervised algorithm, *SkyEx-D*, which does not need a threshold and instead sets the cut-off based on the distance of the skylines. We demonstrate experimentally that *SkyEx-D* can reach a near-optimal F-measure (less than 0.01 loss). Additionally, we offer `skyex`, an R-package that implements the threshold-based and unsupervised skyline-based algorithms, supports the whole entity linkage pipeline with other state-of-the-art methods for entity blocking and comparisons, and provides a powerful Analysis and Visualization module to aid the explainability of the results.

Besides the unsupervised algorithms, we propose a trained skyline-based algorithm, *SkyEx-T*, which is able to learn the preference function and the cut-off in tiny training sets (0.05%-1% of the dataset) and still achieve machine-learning-level accuracy. Moreover, the *SkyEx-T* model is fully explainable and readable, in contrast to the commonly-used black-box machine learning techniques. Furthermore, *SkyEx-T* has no weights nor layered architecture; consequently, it shows high robustness in deployment, while for the machine learning, some re-configuration and re-tuning of parameters might be needed when the new data arrives. Finally, we demonstrate that *SkyEx-T* cut-off closely approximates the optimal cut-off, even though it was learned on a tiny training set. With our algorithms in the spatial entity linkage, we ensure a duplicate-free dataset and rich-in-attribute spatial entities.

Overall, this thesis contributes with effective and efficient algorithms for the initial and fundamental step of every geo-social research study: having recent, good-quality, rich-in-attributes datasets. We propose the *MSSD-** algorithms that make the data extraction process more effective (14.3 times more data than the initial querying) while managing the requests carefully. We further improve the quality of the retrieved data by detecting pairs that refer to the same entity with high precision and recall while having an explainable and robust model (*SkyEx-** algorithms). In the future, in the con-

text of data extraction, we aim to work on hybrid algorithms that combine location-based with user-based and keyword-based API requests and use supervised techniques to learn the parameters of APIs. In the context of spatial entity linkage, we plan to work on hybrid blocking techniques that combine spatial attributes with textual and semantic ones, multi-class classification for the skyline-based algorithms, and crowdsourcing techniques for improving the labeling of the pairs.

Resumé

Webdatakilder indeholder store mængder geosociale data, der består af brugere, venskabs-/tilhængernetværk, check-ins, anmeldelser, placeringer osv., som er af stor interesse for den akademiske verden og for industrien. Der er offentligt tilgængelige datasæt af sådanne webdata, men de er meget gamle (over ti år), ikke store og ikke rige på attributter. Alternativt kan man bruge de offentlige API'er til at få adgang til og downloade webdataen. Desværre er denne proces udfordrende på grund af API'ernes begrænsninger (f.eks. mængden af data hentet i en anmodning, antallet af anmodninger udført inden for en tidsramme osv.). Derfor er der behov for algoritmer, der i betragtning af begrænsningerne er i stand til at hente et datasæt af god kvalitet fra webkilder, et behov, som den aktuelle moderne teknologi ikke imødekommer.

Denne afhandling har til formål at tilvejebringe algoritmer og værktøjer, der kan producere større, nylige, duplikatfrie og rig-i-attribut spatial entitetsdata. For at opnå større og nyere data fra webdatakilder foreslår vi multi-source seed-driven (*MSSD*) algoritmer, der bruger de offentlige gratis API'er til at udtrække geosociale data. *MSSD*-algoritmerne sigter mod at maksimere mængden af data der ekstraheres, samtidig med at antallet af anmodninger minimeres og hver kildes API-begrænsninger respekteres. Ideen bag de seed-drevne algoritmer er at udføre nogle API-anmodninger for at have et indledende datasæt og derefter bruge punkterne af den rigeste kilde som frø i API-anmodningerne til resten af kilderne. Vi foreslår forskellige teknikker til valg af punkter og radius af søgningen. Vi vælger en flerkildeløsning, da flere kilder leverer uafhængig information og forskellige attributter i modsætning til en enkelt kilde. Derudover, demonstrerer vi eksperimentelt, at brug af en enkelt kildealgoritme nogle gange havner i en blindgyde under konvergering. *MSSD*-algoritmer ekstraherer samlet 14.3 gange mere data end den oprindelige forespørgsel, mens optimerede version af *MSSD** henter 90% af dataen med mindre end 16% af anmodningerne i forhold til den ikke-optimerede version.

Når der anskaffes flerkildedata, kan den samme spatiale entitet eksistere i forskellige kilder og nogle gange endda inden for den samme kilde.

Disse "dubletter" er ikke lette at opdage, da de har forskellige attributter, de udtrykkes i forskellige former, og de kan endda indeholde modstridende attributværdier. Problemet med at finde hvilke par spatiale entiteter, der refererer til en reel fysisk enhed, kaldes spatiale entitetsforbindelse. Vi løser dette problem med flere algoritmer, som alle deler den samme geografiske gruppeteknik, og de bruger skylines til at rangordne de sammenlignede par. Den spatiale gruppesteknik (*QuadFlex*), som vi foreslår, er en kvadranttræsinspireret algoritme, der grupperer de spatiale entiteter baseret på afstanden mellem dem og områdets tæthed. Desuden tillader det, at tildelingen af spatiale entiteter i mere end et barn ikke går glip af relevante sammenligninger. De spatiale entiteter, der falder ind i det samme barn bliver sammenlignet parvis.

For at afgøre hvilke par der hører til den samme fysiske entitet, foreslår vi nye skyline-baserede (*SkyEx-**) algoritmer, som bruger præferencefunktioner til at tildele skylines til parrene. De tærskelbaserede *SkyEx*, *SkyEx-F* og *SkyEx-FES* kræver en tærskel, der er antallet af skylines, for at adskille det positive fra den negative klasse, og de er i stand til at opnå et F-measure på 0.72 på hele datasættet og 0.85 på prøver der manuelt er mærket. Vi introducerer en fuldt overvåget algoritme, *SkyEx-D*, som ikke har brug for en tærskel og i stedet indstiller afskæringen baseret på afstanden til skylinerne. Vi demonstrerer eksperimentelt, at *SkyEx-D* kan nå et næsten optimalt F-measure (mindre end 0.01 tab). Derudover tilbyder vi skyex, en R-pakke, der implementerer de tærskelbaserede skyline-baserede algoritmer, som ikke kræver mærkning af data, understøtter hele entitetens koblingspipelinen med andre avancerede metoder til entitet gruppering og sammenligninger, og giver en kraftig analyse- og visualiseringsmodul til at hjælpe med at forklare resultaterne.

Udover algoritmerne der ikke kræver mærket data, foreslår vi en trænet skyline-baseret algoritme, *SkyEx-T*, som er i stand til at lære præferencefunktionen og afskæringen ved brug af små træningssæt (0.05% -1% af datasættet) og stadig opnå en nøjagtighed på niveau af maskinlærings modeller. Desuden er det mulig at forklare, samt læse, resultaterne af *SkyEx-T*-modellen, i modsætning til de almindeligt anvendte black-box maskinlæringsteknikker. Derudover, har *SkyEx-T* ingen vægte eller lagdelt arkitektur; hvilket betyder høj robusthed når den er udrullet, mens maskinindlæringsmodeller muligvis har behov for en vis konfiguration og justering af parametre, når der ankommer nyt data. Endelig demonstrerer vi, at *SkyEx-T*'s cut-off tilnærmer sig den optimale cut-off, selvom det blev lært på et lille træningssæt. Med vores algoritmer i spatiale entitetsforbindelser sikrer vi et duplikatfrit datasæt og med rig-i-attribut geografiske entiteter.

Samlet set bidrager denne afhandling med effektive algoritmer til det indledende og grundlæggende trin i hver geo-social forskningsundersøgelse: at have det seneste, bedste kvalitets, rigest-på-attribut datasæt som muligt. Vi

foreslår *MSSD*-* algoritmer, der gør dataekstraktionsprocessen mere effektiv (14.3 gange flere data end den oprindelige forespørgsel), mens vi håndterer requests omhyggeligt. Vi forbedrer yderligere kvaliteten af de hentede data ved at detektere par, der henviser til den samme enhed med høj præcision og tilbagekaldelse, mens vi har en forklarbar og robust model (*SkyEx*-* algoritmer). I fremtiden tilstræber vi i forbindelse med dataudvinding at arbejde på hybridalgoritmer, der kombinerer placeringsbaseret med brugerbaserede og søgeordsbaserede API-anmodninger og bruger overvågede teknikker til at lære parametrene for API'er. I forbindelse med spatiale entitetsforbindelser, planlægger vi at arbejde på hybridgruppeteknikker, der kombinerer spatiale attributter med tekstlige og semantiske, klassificering i flere klasser for de skylinebaserede algoritmer og crowdsourcing-teknikker til at forbedre mærkningen af par.

Resumé

Les sources de données Web contiennent de grandes quantités de données géo-sociales, constituées d'utilisateurs, de réseaux d'amitié/d'abonnés, d'enregistrements, de commentaires, de lieux, etc., qui sont d'un grand intérêt pour le milieu universitaire et l'industrie. Il existe des jeux de données accessibles publiquement contenant des échantillons de ces données Web, cependant, ils sont souvent anciens (plus de dix ans), peu volumineux, et pauvres en attributs. Il est également possible d'utiliser des API publiques pour accéder aux données Web et les télécharger. Malheureusement, ce processus est difficile en raison des limitations des API (par exemple, la quantité de données récupérable lors d'une requête, le nombre de requêtes effectuable par unité de temps, etc.). Par conséquent, il y a un besoin d'algorithmes qui, compte tenu des limites, sont capables de récupérer un ensemble de données de bonne qualité à partir de sources Web, un besoin que l'état actuel de l'état de l'art ne peut satisfaire.

Cette thèse vise à fournir des algorithmes et des outils capables de produire des données d'entités spatiales plus volumineuses, récentes, sans doublons, et riches en attributs. Pour obtenir des données plus volumineuses et récentes à partir de sources de données Web, nous proposons des algorithmes multi-sources basés sur les semences (*MSSD*) qui utilisent les API publiques gratuites pour extraire des données géo-sociales. Les algorithmes *MSSD* visent à maximiser la quantité de données extraites tout en minimisant le nombre de requêtes et en respectant les limitations de l'API de chaque source. La raison d'être des algorithmes basés sur les semences est d'exécuter certaines requêtes API pour avoir un ensemble de données initial, puis d'utiliser les points de la source la plus riche comme semences dans les requêtes API pour le reste des sources. Nous proposons différentes techniques pour choisir les points et le rayon de la recherche. Nous optons pour une solution multi-sources étant donné que plusieurs sources fournissent des informations indépendantes et des attributs divers au lieu d'utiliser une seule source. De plus, nous démontrons expérimentalement que l'utilisation d'un algorithme à source unique converge parfois vers une impasse. Les algorithmes *MSSD* extraient en moyenne 14.3 fois plus de données que la requête

initiale, et la version optimisée *MSSD** récupère 90% des données avec moins de 16% des requêtes de la version non optimisée.

Lors de l'obtention de données multi-sources, la même entité spatiale peut exister dans différentes sources et parfois même au sein de la même source. Ces "doublons" ne sont pas faciles à détecter car ils ont des attributs différents, ils sont exprimés sous différentes formes et ils peuvent même contenir des valeurs d'attributs contradictoires. Le problème de trouver quelles paires d'entités spatiales se réfèrent à une entité physique réelle est appelé liaison d'entités spatiales. Nous abordons ce problème avec plusieurs algorithmes qui partagent tous la même technique de blocage spatial et qui utilisent des skylines pour classer les paires comparées. La technique de blocage spatial (*QuadFlex*) que nous proposons est un algorithme inspiré du quadtree qui regroupe les entités spatiales en fonction de la distance qui les sépare et de la densité de la zone. De plus, il permet d'attribuer des entités spatiales à plus d'un enfant afin de ne manquer aucune comparaison pertinente. Les entités spatiales appartenant au même enfant sont comparées par paires.

Pour décider quelles paires appartiennent à la même entité physique, nous proposons de nouveaux algorithmes basés sur des skylines (*SkyEx-**) qui utilisent des fonctions de préférence pour attribuer des skylines aux paires. Les algorithmes basés sur des seuils comme *SkyEx*, *SkyEx-F* et *SkyEx-FES* nécessitent un seuil correspondant au nombre de skylines pour séparer la classe positive de la classe négative. Ces algorithmes sont capables d'atteindre une F-measure de 0.72 sur l'ensemble de données et de 0.85 sur un échantillon étiqueté manuellement. Nous introduisons un algorithme entièrement non supervisé, *SkyEx-D*, qui n'a pas besoin de seuil et définit à la place la coupure en fonction de la distance des skylines. Nous démontrons expérimentalement que *SkyEx-D* peut atteindre une F-measure quasi optimale (moins de 0.01 perte). De plus, nous proposons skyex, un package R qui implémente les algorithmes basés sur des seuils et des skyline non supervisés, qui prend en charge l'ensemble du pipeline de liaison d'entités avec d'autres méthodes de pointe pour le blocage et les comparaisons d'entités, et qui fournit un module d'analyse et de visualisation puissant pour faciliter l'explicabilité des résultats.

En dehors des algorithmes non supervisés, nous proposons un algorithme entrainé basé sur des skylines, *SkyEx-T*, capable d'apprendre la fonction de préférence et le seuil dans de minuscules ensembles d'apprentissage (0.05% - 1% de l'ensemble de données) et qui atteint toujours une précision du niveau des méthodes d'apprentissage automatique. De plus, le modèle de *SkyEx-T* est entièrement explicable et lisible, contrairement aux techniques d'apprentissage automatique de black-box couramment utilisées. De plus, *SkyEx-T* n'a ni poids ni architecture en couches ; par conséquent, il montre une grande robustesse dans le déploiement, tandis que pour l'apprentissage

automatique, une reconfiguration et un réajustement des paramètres peuvent être nécessaires lorsque de nouvelles données arrivent. Enfin, nous démontrons que le seuil de *SkyEx-T* se rapproche étroitement du seuil optimal, bien qu'il ait été appris sur un petit jeu d'entraînement. Avec nos algorithmes de liaison d'entités spatiales, nous garantissons un jeu de données sans doublons et des entités spatiales riches en attributs.

Dans l'ensemble, cette thèse propose des algorithmes efficaces et efficients pour l'étape initiale et fondamentale de chaque étude de recherche géo-sociale: disposer d'ensembles de données récents, de bonne qualité, et riches en attributs. Nous proposons les algorithmes *MSSD*-* qui rendent le processus d'extraction de données plus efficace (14.3 fois plus de données que la requête initiale) tout en gérant les requêtes avec soin. Nous améliorons encore plus la qualité des données récupérées en détectant les paires qui se réfèrent à la même entité avec une grande précision et rappel tout en ayant un modèle explicable et robuste (algorithmes *SkyEx*-*). À l'avenir, dans le contexte de l'extraction de données, nous visons à travailler sur des algorithmes hybrides qui combinent des requêtes d'API basées sur la localisation, l'utilisateur, et des mots-clés, en les combinant avec l'utilisation de techniques supervisées pour apprendre les paramètres des API. Dans le contexte de la liaison d'entités spatiales, nous prévoyons de travailler sur des techniques de blocage hybrides qui combinent des attributs spatiaux avec des attributs textuels et sémantiques, la classification multi-classes pour les algorithmes basés sur des skylines, ainsi que des techniques de crowdsourcing pour améliorer l'étiquetage des paires.

Acknowledgements

Completing this journey was challenging and I was able to achieve it thanks to several wonderful people that stayed by my side.

I would first like to express my gratitude to my PhD supervisors for leading me through this journey. I would like to thank Prof. Torben Bach Pedersen for his essential role in helping me formulate research problems, discuss solutions, and give timely feedback. Thank you for always finding time for me! I always felt like I had you by my side in this journey, and that feeling is what kept me going on. Thank you for being understanding, supportive, and a good friend. I would like to thank my co-supervisor, Prof. Esteban Zimányi, for believing in me since six years ago, when I received the scholarship for the Erasmus Mundus IT4BI Master. I still remember how happy I was to get that acceptance email signed "Esteban Zimányi" in the end! I never thought back then that my professor of "Advanced Databases" would be my collaborator one day. I am grateful for your support, brainstorming meetings, and friendly advises. Special thanks go to my other collaborators Giorgos Giannopoulos and Vassilis Kaffes, for contacting me and expressing their interest in my work. I extremely enjoyed our discussions and especially the technical, transparent, and intelligent talks with Vassilis. Finally, I would like to thank the AAU and ULB jury of Prof. Manfred, Prof. Konstantinos, Prof. Maria Luisa, and Assoc. Prof. Mahmoud for accepting to be part of my committee and taking the time to read my thesis. I am looking forward to meeting you and having interesting discussions.

During this PhD, I have been lucky to meet lovely people that have supported me and filled my life with joy on grey days. I would like to thank Olga, Felipe, and Davide for being amazing friends and making my days beautiful. After you left, I could feel a deep emptiness in my heart. I am pleased to have met Bhuvan, Aamir, Bijay, Aftab, Ilkcan, Simon, Tobias, Emil, Alvis, Theis, Jilin, Kasper, Kashif, Daniel, Christian, Jonas, Nuref, and Van and share good times together. I would like to thank Olivier, Carlos, Roshni, Matteo, Rudra, and Nguyen for their positive spirit, especially for lifting me up during the corona lockdown. I will miss our casual gathering, the good food, the funny jokes, and the dancing. I wish to express my gratitude

to Theis and Olivier for helping me translate the abstract into Danish and French, respectively.

During my external stay in Université libre de Bruxelles, I was fortunate to meet Faisal, Thiago, Azadeh, Hiba, Idris, Jean, Jean-Philippe, Gilles, Judith, Waqas, and Bakli, who made my time in Brussels memorable. Thank you for being friendly and making me feel welcome! I was happy to reunite with Moditha, the closest person to my heart for the last six years and become again inseparable. Since the first day we met (September 8th, 2015, at 365 Avenue de la Couronne, Bruxelles), you have been there for me, and I cannot think of another person that could give me the strength you do for continuing forward. I need to express my thankfulness to my lovely friends Jorge, Gledys, Katya, Anas, Larissa, Ward, Victor, Juan, Paula, Estel, Krista, Romina, Katerina, Ines, Laura, Elena, Ziyad, and Thao for their kindness, support, and pure love.

Special thanks go to Charlotte and Vinciane from ULB, and Susanne, Ulla, Lene and both sweet Helles from AAU for their effective and efficient solutions on administrative matters and their beautiful smiles that always lifted me up. I would like to express my appreciations to the IT4BI-DC committee for giving me the opportunity to be part of the IT4BI-DC experience and evaluating my progress in the summer schools. I enjoyed presenting my work and getting to know the people behind this excellent program. Finally, I would like to thank all my colleagues at AAU and ULB for every moment that their presence has facilitated this journey.

I deeply thank my family for their unconditional love. I want to thank my sister, Ledia, and my brother, Gjergji, for their good vibes and their positive energy, always bringing colors to my day. I thank my parents, Shuaip and Teuta, for their understanding, good humour, and continuous support in every step of this journey. I want to thank my grandmother, Roza, for always being on my side, even when I am wrong, and for her sweet grandmotherly compliments. I would like to remind my sweet grandmother Zenepe, who passed away a bit more than a year ago, of her joyful spirit, funny jokes, and immense love. It is not the same without you. Additionally, my thanks go to both my aunts, my uncle, my close cousins, especially Mirel, who always checks on me and encourages me.

I want to thank Denmark and Belgium for welcoming me, teaching me new recipes, and adding a part of their culture to my personality. Finally, I thank the universe for the powerful good energy, and I hope that with my expressed gratitude, I am sending back some goodness!

Contents

Thesis Details

Thesis Title:	Multi-Source Spatial Entity Extraction and Linkage
Ph.D. Student:	Suela Isaj
Supervisors:	Prof. Torben Bach Pedersen, Aalborg University
	Prof. Esteban Zimányi, Université libre de Bruxelles

The main body of the thesis consists of the following papers.

[A] S. Isaj, T. B. Pedersen. "Seed-Driven Geo-Social Data Extraction". In: *Proceedings of the 16th International Symposium on Spatial and Temporal Databases (SSTD)*, ACM, 2019.

[B] S. Isaj, T. B. Pedersen, E. Zimányi. "Multi-source spatial entity linkage" In: *IEEE Transactions on Knowledge & Data Engineering (TKDE)*, IEEE, 2020

[C] S. Isaj, T. B. Pedersen. "skyex - an R Package for Entity Linkage". In: *Proceedings of the 23rd International Conference on Extending Database Technology (EDBT)*, ACM, 2020 .

[D] S. Isaj, V. Kaffes, T. B. Pedersen, G. Giannopoulos. "Explainable and Robust Skyline-Based Spatial Entity Linkage on Tiny Training Data". Under submission

In addition to the main papers, Paper E has also been included. Paper B is the extended version of Paper E, thus Paper E can be considered as a subset of Paper B. Paper E was awarded **"Best Paper Award Runner-Up"** in *Proceedings of the 16th International Symposium on Spatial and Temporal Databases (SSTD'19)*, Vienna, Austria, August 2019.

[E] S. Isaj, E. Zimányi, T. B. Pedersen. "Multi-source spatial entity linkage" In: *Proceedings of the 16th International Symposium on Spatial and Temporal Databases (SSTD)*, ACM, 2019.

This thesis has been submitted for assessment in partial fulfilment of the PhD degree. The thesis is based on the submitted or published scientific papers, which are listed above. Parts of the papers are used directly or indirectly in the summary of the thesis. As a part of the assessment, co-author statements have been made available to the assessment committee and those are also available at the at the Technical Faculty of IT and Design at Aalborg University and the Department of Computer and Decision Engineering at Université libre de Bruxelles. The permission for using the published articles in the thesis has been obtained from the corresponding publishers with the conditions that they are cited and DOI pointers and/or copyrights/credits are placed prominently in the references.

Suela Isaj
Aalborg University, March 10, 2021

Part I

Thesis Summary

Multi-Source Spatial Entity Extraction and Linkage

1 Introduction

This thesis focuses on spatial entities, which are geo-located and fully identified entities that can be found in multiple location-based sources. We will specifically address the problem of spatial entity extraction, and later, we will discover which of these entities belong to the same physical entity. We will first start with what inspired and motivated this thesis; then, we will continue with the objectives and the scope of the thesis, and finally, we will introduce the structure of the thesis content.

1.1 Background and motivation

The web is a hospitable environment for various sources to publish data. Not only is the volume of the data growing, but the heterogeneity of the information is also increasing. An attractive type of data is geo-related data, usually found in the form of spatially located entities, geo-tagged user-content, etc. When combined with users and user connections, this data is the core of geo-social research. Examples of interesting and contemporary geo-social topics are mobility analytics [1–4], influential users or locations [5–7], geo-related recommender systems [8–13], etc. Despite the popularity of this data, the main focus has only been how to perform research with it rather than how to obtain and enrich it.

We investigated the origin of the geo-social data used in published articles in the field, and we noticed that most of this data comes from publicly available datasets. However, most of this data belongs to location-based social networks like Gowalla and Brightkite, which are not operational for almost 10 years now, and their data is considerably old. We show this phenomenon in Fig. 1.1, where the x-axis represents the year of the published article and the y-axis the latest year in the dataset. The points in the graph are refer-

ences to 51 geo-social research papers. Ideally, these points would position themselves on the diagonal, which means that the articles use current data. Around 60% of these papers use a dataset more than 3 years old, while for around 40% of them, this gap goes to above 7 years. The issue with older datasets is that not only do we study outdated phenomena, but the reference to the current reality cannot be established. Besides, these datasets are not dense enough in check-ins, and given that the activity is old, we cannot enrich them with data from currently-operational data sources. For example, check-ins from 7 years ago might not belong to today's same geo-located place, thus enriching them with details from today's Google Maps data might not be correct. Finally, given the growth of web data, it is unfortunate that most geo-social research chooses the publicly available datasets rather than benefiting from the richness of the web data sources.

Alternatively to using publicly available datasets, one could extract data from web sources. Most of the location-based sources offer APIs, which, in contrast to web scraping, are well-structured, easy to use, and the authorized way to extracted data. However, the APIs have several limitations in order to control and regulate the amount of data extracted. Therefore, *the geo-social data extraction algorithms in this thesis were strongly motivated by the need for new, rich, and heterogeneous data and the lack of effective and efficient data extraction algorithms that can deal with different location-based sources*.

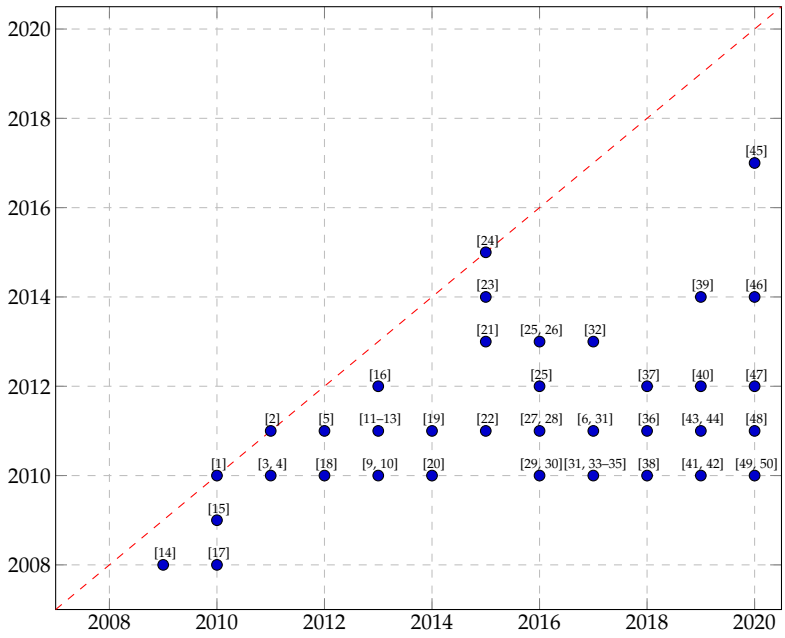


Fig. 1.1: The year of the published article (x-axis) versus the latest year in their dataset (y-axis)

For extracting geo-related data, we will need more than one data source. Firstly, no source is complete, meaning that it contains all the spatial entities in a specific area. Thus, if we want to obtain a near-complete set of spatial entities in an area, we will have to extract data from different sources. Secondly, even if there were a source with the whole set of spatial entities, a multi-source solution would still be preferred because we do not want a single source with the "monopoly" of information; rather, we value information independence transparency. Finally, each data source provides diverse attributes, heterogeneous information, and this is needed to have a multi-dimensional representation of each spatial entity. Consequently, *a multi-source solution is needed and preferred to obtain a larger number of spatial entities, with diverse attributes, originating from independent sources*.
The main type of data extracted from location-based sources is *spatial entities*. Since multiple sources will be needed for geo-related data extraction, different records of the same physical spatial entity will co-exist. Additionally, even within the same source, there might be duplicate information about the same entity. Solving this issue can be as easy as finding exact duplicates and as complicated as needing a trained algorithm to evaluate if two records belong to the same entity. This problem is usually referred to as *entity linkage* and in our case, as *spatial entity linkage*. *While the entity linkage has been addressed in several works, the spatial entity linkage has not yet been given a considerable amount of attention*.
This thesis proposes algorithms that use multiple sources for spatial entity extraction and linkage. The need for these algorithms is notably observed when looking at the current geo-social research papers and their need for fresh and rich data. We address the demand for effective and efficient algorithms to obtain a larger multi-source [geo-social data extraction] and rich-in-attributes, duplicate-free dataset of spatial entities [spatial entity linkage].

1.2 Objectives of the thesis

The aim of this thesis is that *in the context of heterogeneous, multiple-origin, location-based web data, to provide effective and efficient algorithms for obtaining a good quality spatial entity dataset in terms of the amount of data, free of duplicate records, and rich in attributes*. In order to achieve this aim, we fulfil the following objectives:

1. Provide effective location-based data extraction algorithms for obtaining geo-related data from location-based sources. The algorithms should be general enough to adapt for multiple sources and at the same time, specific enough to accommodate the API limitations of each source. The algorithms' goal is to maximize the amount of data extracted while minimizing the number of requests, thus being both ef-

fective and efficient.

2. Propose accurate and explainable algorithms for linking records of spatial entities that belong to the same physical entity. Given that more than one source is needed to obtain a rich set of spatial entities, the algorithm should be suitable for multi-source data, containing duplicates even inside the same source. Moreover, the algorithms should overcome the current obstacles of lack of labeled data, and the need for model explainability and robustness in deployment. Additionally, we offer a user-friendly tool for entity linkage that can be easily integrated with other data preparation and data mining programming environments.

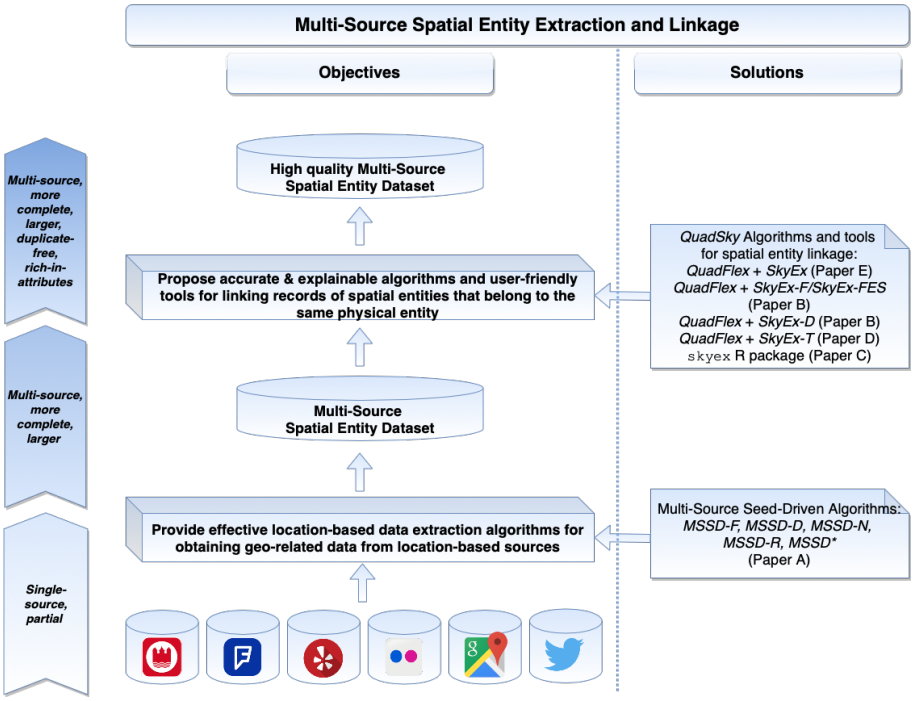


Fig. 1.2: Objectives of the thesis and the proposed solutions

We address these objectives in five papers [51–55], as shown in Fig. 1.2. For each objective on the left side of the figure, we propose the solutions on the figure's right side. We start from the bottom, with several location-based sources. Extracting data from each of these sources would yield the default single-source partial set of extracted spatial entities. Our proposed multi-source seed-driven algorithms (*MSSD*) can extract a larger set of spatial entities for the same requests, resulting in a more complete set, originating from

several sources. To improve the data quality of the extracted data, we need to detect the duplicates. We propose a spatial blocking technique (*QuadFlex*) and the *SkyEx-** algorithms that can solve the spatial entity linkage problem. As a result, we obtain a larger, more complete, duplicate-free, rich-in-attributes multi-source set of spatial entities.

1.3 Structure of the thesis

The thesis summary is composed of 7 sections. We introduce state of the art for spatial entity extraction and spatial entity linkage research papers in Section 2. The geo-social data extraction problem is formulated in Section 3, and then, we propose the seed-driven algorithms for data extraction from multiple geo-related sources. Section 3's content is based on Paper A [51]. Section 4 explains the problem of spatial entity linkage and our overall proposed solutions. Then, we introduce our spatial blocking algorithm, which is used in Paper B [53], D [55], and E [52]. Later, we detail our threshold-based (*SkyEx*, *SkyEx-F*, and *SkyEx-FES*) and unsupervised (*SkyEx-D*) skyline-based algorithms which label the pairs based on the likelihood of being the same physical spatial entity. Finally, we explain the main components and functionalities of the `skyex` R package that implements mainly the skyline-based algorithms and also covers the whole entity resolution pipeline with some common algorithms and methods from the literature. Section 4 is based on papers B [53], C [54], and E [52]. In Section 5, we propose a trained skyline-based algorithm (*SkyEx-F*), which uses a tiny training set and still achieves machine learning-level accuracy, while being explainable and robust in deployment. We summarize our contributions in Section 6 and consider future directions in Section 7.

2 State of the art

This section will describe the related work on geo-social data extraction and spatial entity linkage. We will start with common techniques that have been used so far to extract geo-social data from social networks, location-based social networks, and location directories. We will then discuss the data integration process, the general entity linkage problem, and finally, we will focus on the spatial entity linkage research.

2.1 Geo-social data extraction

The *geo-social* data includes geographical data, user data (usually in the form of user profiles), social relationships between users, and relationships between users and locations (usually as check-ins). This type of data has led

to various research topics, such as new geo-social metrics [15], crowd pattern mining and anomaly detection [1, 2], explaining social ties based on check-in activity [3, 4], studying influential users or locations [5–7], recommender systems [8–13], etc. However, even though most geo-social data sources provide APIs to allow data extraction, the number of requests are limited and are subject to different restrictions, depending on the source. *Despite the popularity of the geo-social data, there is no related work dedicated specifically to geo-social data extraction*. However, most of the related works indirectly mention their data extraction process, which can be categorized as *user-based*, *location-based*, and *keyword-based*.

2.1.1 User-based crawling

The user-based crawling navigates the geo-social data source using users as query parameters. The most popular method mentioned in several papers is *Snowball* [15, 21, 56]. *Snowball* starts with some initial users, known as the *seed*. After querying with the seed, Snowball stores the data of the seed users (profiles, attributes, check-ins, reviews, locations they visited) and then expands to their network (friends, followers, etc.) and uses them as the next query parameters. Thus, each time, the seed grows considerably, providing more and more data.
However, there are some drawbacks to this method. First, *Snowball* is biased to the high degree nodes [57]. Second, the whole process depends highly on the initial seed, so it needs to be selected carefully. Finally, Snowball is not very effective on data sources that have their main focus on location rather than users (Yelp, Google Places), because their users' social networks are either small or nonexistent.
Besides the Snowball, several works have used *linked accounts* [16, 27, 45, 58], which are users that have declared their corresponding accounts on different social networks, usually one account on a typical social network such as Twitter or Facebook, and the other in a more location-based data source such as Foursquare, Brightkite, Yelp, etc. These accounts provide a holistic view of the user's geo-social activity, covering his profile characteristics, networks of friends, check-ins, location history, etc. Nevertheless, such accounts are rare to find (less than 1%).

2.1.2 Location-based crawling

The location-based crawling uses locations as query parameters, usually a point and a radius, or a bounding box. In contrast to the user-based crawling where we are supposed to provide some initial user ids, we can immediately start querying with the area of interest. The result of the query will be places and locations in the queried area, their details (name, address, phone), check-

ins and reviews. However, after querying with the same area of interest, we might keep getting the same data. To get out of this loop, Lee et al. [1] query Twitter periodically, starting first with some initial points. Then, after some time window, they use the previous step's points as the new query points. Hence, they can detect new locations indicated by the users in each step, without wasting the requests on empty areas. We will use the term *Self-seed* to refer to this method. In each point of time *n*, *Self-seed* uses the points discovered in step *n-1* as query points for the step *n*.

2.1.3 Keyword-based crawling

The keyword-based crawling uses keywords as query parameters. Typically, this type of crawling has not been used for location-based data, but rather on works on text mining, opinion mining, named entities, constructing the reputation of an entity, etc. [59–61]. However, querying with the location's name might return data that is geo-located in the area indicated by the keyword, but the results might also contain irrelevant data. For example, querying with "Brussels" might return the city of Brussels, places with "Brussels" in their name (e.g. "Brussels Fries", "Museum of Brussels", etc.) that might be or not be in Brussels, texts related to Brussels, and maybe even texts related to brussels sprouts.

2.2 Spatial entity linkage

The spatial entity linkage problem aims to find those pairs of entities originating from the same or different sources that refer to the same spatial entity. In this section, we will study first the related work regarding the data integration, we will reformulate the general problem of entity resolution, we will mention some advances in the field of geographical data integration, and finally, we will study in detail the works of spatial entity linkage, which are the most related to our problem.

2.2.1 Data integration process

The process of combining data originating from different sources, and obtaining a holistic view of the information is usually referred to as *data integration* [62]. This process can be as simple as finding exact duplicates to dealing with big amounts of heterogeneous and dynamic data. Thus, in order to be able to solve this task, data integration needs three steps, which are schema alignment, record linkage, and data fusion [63].
The *schema alignment* is working through the semantics of attributes of each source and matching them to attributes of the other sources. The schema alignment produces three outputs: a mediated schema, attribute matching,

and a schema matching. The mediated schema provides a view of all the sources and the domain, the attribute matching maps the attributes of each source to the corresponding ones in the other sources, and the schema matching connects each source schema to the mediated schema considering the semantics relationships between the contents. After the schema matching, the next step is the *record linkage*. In this step, we study the record and its attributes, and we try to partition the data so that each partition refers to the same real entity. Finally, the *data fusion* merges the records in a partition into one. In this thesis, we work with spatial entities which originate from relatively well-structured sources such as location-based social networks (Foursquare), or location directories (Krak, Google Places). Thus, the attributes do not contain semantic ambiguity; the schema is rather simple, so we do not perform schema alignment. Furthermore, the data fusion is not within the thesis's scope, but rather an interesting future work direction. Therefore, the next sections will be focused on the second step of data integration, the record linkage.

2.2.2 Entity linkage

Entity linkage aims at finding different descriptions of the same real entity within the same or across sources [64]. Similar synonyms describing the same problem have continuously appeared in the literature such as *deduplication*, *entity resolution*, *entity matching*, *record linkage* [63, 65]. The entities that are matched can be of various fields, for example, profiles in social networks belonging to the same individual [56, 66], bioinformatics data [67], biomedical data [68], publication data of the same author [65, 69], genealogical data to find the human entities [70], records of the same product [65, 69], etc. Regardless the field, the entity linkage follows, in principle, three main steps: blocking, entity comparison, and pair labeling [54, 71] (Fig. 1.3). In some cases, blocking processing can be an extra step after the blocking in order to remove redundant comparisons [64, 72, 73].

Blocking. Blocking is grouping entities that are somehow similar and worth comparing further. Since a Cartesian product of all the possible comparisons is unfeasible in the era of big data, the goal of blocking is to reduce the number of comparisons, while ensuring that we are not missing real matches. Hence, blocking is not focused on exact comparisons, which tend to be ex-

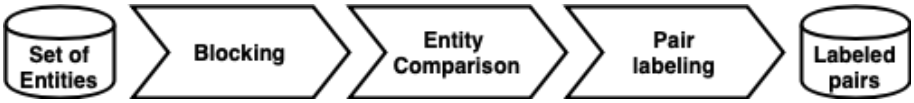


Fig. 1.3: The entity linkage process (Reproduced from [54])

pensive, but rather fast and approximate grouping. Attribute clustering creates non-overlapping clusters based on the selected attribute [74–77]. Some blocking techniques are based on shared tokens, q-gram blocking and prefix, suffix and infix blocking [73, 74, 76, 77]. These blocking methods are mostly based on grouping the records based on the similarity of the textual attributes and do not deal with coordinates. However, there are attempts to use these token-based blockings on web data with a spatial flavor [77] but only 39%-63% of links we discovered, which means that we would lose a big part of the matches before even comparing the entities. Hence, this section's blocking techniques are not suitable for spatial entities.

Entity comparison and pair matching. After having the blocks, we can compare the entities within a block. The entities are compared based on their attributes using similarity metrics. To decide which entities are similar enough to be matched, some works propose thresholds for each attribute similarity or combine them in a similarity scoring function [78–80]. However, finding a suitable threshold might be challenging and require extensive experiments. Therefore, some related works use a classifier to learn a good scoring function [81–83]. However, sometimes the decision to match two entities might contain uncertainty, and we can express this uncertainty in the matching process [84]. Finally, the entity linkage process might need some human feedback to improve the accuracy, so, within a predefined budget, we can include users or an oracle [65, 69, 69].

2.2.3 Geographical data integration

First, we need to identify the differences between a *spatial object* and a *spatial entity*. A spatial object is entirely identified by the spatial characteristics such as longitude and latitude, or geographical shape. In contrast, a spatial entity is a fully identified entity with attributes such as a name, a phone number, photos, etc., which is geo-located. In a typical scenario, a spatial entity contains a spatial object. However, two spatial entities can share the same spatial object, for example, two businesses in the same building but different floors, and also a spatial entity might contain two spatial objects, for instance, an amusement park having two theme parks lying in different spatial objects (Disneyland Park and Walt Disney Studios Park in Disneyland Paris). Hence, the data integration of spatial objects is, in essence, different from the spatial entity linkage.
The research on geographical data integration is mostly focused on matching spatial objects, having a unified representation from multiple sources. Road network integration is tackled by Schafers et al. [85] where rules are used to find roads that are the same or not. The roads' similarity is calculated in terms of the length, angles, shape, and sometimes the name of the street

is used, resembling more to a spatial entity linkage problem. Nonetheless, this approach is tailored specifically for roads and not applicable to spatial entities. Similarly, [86–89] propose purely spatial solutions and match spatial objects from sensors and radars. These solutions aim to create a surface representation in 2D or even in 3D from the unified spatial objects.

2.2.4 Spatial entity linkage

Spatial entities lie in between spatial object and entities. The spatial entity linkage process follows the same steps as entity linkage: blocking, entity comparison, and pair labeling.

Spatial entity blocking. Besides the blocking techniques in [73, 74, 76, 90] that are not applied to spatial entities, there are some trivial spatial blocking techniques used in [80, 91]. In [80], the spatial blocks are created using spatial entities at most 5 m apart. This threshold is user-defined without providing reasoning behind this choice. In contrast to [80], the threshold of the spatial distance in [91] is defined based on the type of spatial entities, small thresholds of 50 meters for bars and restaurants, but 500 meters for open spaces like parks. However, both works do not propose an automatic solution for spatial blocking, but rather offer arbitrary user-defined thresholds.

Spatial entity comparison. The spatial entities have attributes such as name, categories, etc. For measuring these similarities, the traditional similarity metrics such as Levenshtein, Jaccard, Cosine [56, 80, 83, 91] show good results. However, more advanced metrics can better capture the similarity of two entities such as a Soft-TFIDF with Levenshtein [92], and traditional string similarities trained further with supervised machine learning [93]. Even though these metrics are not tailored for spatial entities, they have shown promising results when applied to spatial entities [80, 91]. Nevertheless, there are proposed solutions specifically designed for the name attributes of spatial entities which increase the accuracy of the matching process [94–98].
The work in [94] proposes a hybrid method combining token-based and edit-based approaches for street names, while in addition to these, in [95], the accentuation and other language-specific aspects are taken into account when comparing toponym names. In [99], the significant words (core terms) in the spatial entity name are identified, and then, they use unsupervised learning to combine it with a spatial context model. Despite the attempts to improve the spatial entity comparison with better similarity metrics, the results do not significantly improve. The significant increase in the accuracy is achieved only when using deep network model architectures [100, 101]. Alternatively, a meta-similarity function that uses domain-knowledge combined with clas-

sifiers is proposed in [96, 98], which achieves high accuracy without the need for training a deep network.
In addition to the name similarity, some of the works try to capture the semantic similarity of the entities [91] or learn it with a training set [102].

Spatial entity pair labeling. In this stage, considering the similarities of two entities, we need to label the pairs that refer to the same spatial entity as positive and the rest as negative. The work in [99] uses only the names of the places and an unsupervised language model to capture the name and domain knowledge. The solutions proposed in [96, 100, 101, 103] match toponyms based only on their names as well, and the decision of the labeling is based on the outcome of the machine learning or deep learning model. However, all these solutions match toponyms, which are generally names of places or cities and sometimes are not even geo-located, differing significantly from a spatial entity.
Sehgal et al. [102] use a classifier to learn the weights in the similarity scoring function and label the pairs. However, this work is between spatial objects and spatial entities because the entities have names, coordinates, and types like an entity but they are more similar to spatial objects referring to landscapes (rivers, deserts, mountains, etc.). Berjawi et al. [104] combine all the attribute similarities as an average in a similarity score. Alternatively, in [91] $\frac{2}{3}$ of the weight is carried by the name, the geodata and the type of the spatial entity, given that these attributes are always present, while $\frac{1}{3}$ is assigned to the website, the address and the phone number. Morana et al. [91] uses the belief theory [105] to label the pairs.

2.3 Comparison to thesis contributions

The geo-social data has inspired several research topics, but so far, how to retrieve this data has not been one of them. The geo-social data extraction, superficially mentioned in some works, uses either users, locations, or keywords. The keyword-based querying is not suitable for geo-social data extraction because the result might contain a considerable amount of irrelevant information. The methods that use user-based crawling need an initial seed of users and are limited only to user-focused data sources. The work in [1] is similar to ours since we also refine the next query parameters based on the previous step's points. However, there are significant differences: (i) we are not limited to discovering points in only one source; instead, we use different sources as *seed* in each step, (ii) we adapt the other parameters of the query as to maximize the data retrieved while minimizing the number of requests, while this problem is not considered at all in the case of *Self-seed* (iii) our method does not need long waiting before the next query, thus, yielding

faster geo-social data extraction. Overall, *our proposed solution is the first to address the problem of optimizing the geo-social data extraction using APIs, is generic enough to be applied to multiple sources simultaneously, and in the same time, is specific to accommodate each source's requirements.*
As for the spatial entity linkage, in contrast to the entity resolution field that has shown significant advances, spatial entities have not yet received significant attention. The blocking techniques are usually based on textual attributes, and the few spatial blocking methods are based on user-defined radius thresholds. *We address this concern by contributing with a flexible spatial blocking, QuadFlex, which takes inspiration from a quadtree but allows the intersection of the partitions as not to miss any comparisons*. Moreover, *QuadFlex* adapts the radius based on the area's density, which is what [91] attempted to achieve by manually setting different radiuses based on the type of spatial entity.
The solutions proposed in [94, 96, 99–101] for the pair comparison and labeling are for toponyms, which are simply names of places and not multi-attribute geo-located spatial entities. However, their advances on the name comparisons show good accuracy, but sometimes at the cost of designing a deep network architecture. Differently, the spatial entity linkage solutions [80, 91, 104] use traditional string metrics and semantic comparisons for comparing spatial entities, which comes at a small cost but with lower accuracy. As for the pair labeling, we propose three novel solutions, all based on skyline rankings of the pairs: *SkyEx-F*, *SkyEx-D*, and *SkyEx-T*, which were inspired by the *SkyEx* algorithm [52]. Instead of having a similarity score that needs weights [80, 104] or learn the weights with a classifier [102], we use preference functions that connect all the similarity score with the Pareto function. *In contrast to all related work in spatial entity linkage and even entity resolution, a skyline-based solution is used here for the first time to label the pairs*. Differently from [101–103] *SkyEx-F* and *SkyEx-D* do not need training: *SkyEx-F* is a threshold-based solution, and *SkyEx-D* is an unsupervised technique that separates the classes based on the density in the skylines. *SkyEx-T* uses a tiny training set, and the meta-similarity function proposed in [96, 103] to capture better the similarity of spatial entities and learn a preference function that better fits the data. *SkyEx-T demonstrates that there is no need for heavy and deep networks nor large training set, as in [100, 101], to learn a model for spatial entity linkage*.

3 Geo-social data extraction

In this section, we will define the problem of geo-social data extraction and propose five algorithms that can efficiently address the problem while maximizing the extracted data. More details on this problem, our proposed solu-

tions and more extensive experiments can be found in Paper A [51, 106].

3.1 Optimizing geo-social data extraction

We will denote the sources that contain geo-social data as S. The geo-social data extracted from these sources is highly heterogeneous; sometimes it is in the form of a tweet, with a text, user account, geo-located at a point, etc., sometimes in the form of a geo-located photo with a caption, sometimes as a business located at a point and accompanied by user reviews, etc. In order to have a general representation of this data, we use a location-centred definition, as in [51]:

Definition 1.1. *A location l is a spatial entity identified within the source by a unique identifier $id(l)$. A location l has a set of attributes $A = \{a_1, a_2...a_n\}$ accompanied by their values $\{a_1(l), a_2(l)...a_n(l)\}$. A required attribute for a location l is its geographical coordinates denoted as $p(l)$.* (Reproduced from [51])

Hence, these sources provide locations, which are identified by the attributes in the respective sources. Each source S offers an API for data extraction, which, in a general setting, is used by querying with a point p and radius r, and the result of the query is the locations L_p^r in the $Circle(p, r)$. The goal of the geo-social data extraction is to query the source S with its API, using the parameters in such a way, that the number of locations retrieved is maximal for a specific number of requests. We use the problem definition as in [51]:

Problem definition: Optimizing geo-social data extraction is the problem that given a source S_i and a number of requests n finds the sequence of pairs of point and radius $\{< p_1, r_1 >, < p_2, r_2 > ... < p_n, r_n >\}$ such that the size of $L_i = \bigcup_{j=1}^{n} L_{p_j}^{r_j}$ is maximized. (Reproduced from [51])

3.2 Seed-driven algorithms

Given that we have no prior knowledge on the distribution of the locations in each source, it is challenging to find effective combinations of p and r that can maximize the retrieved L_p^r. However, given that all the sources refer to the same physical world, dense areas in data in one source might indicate the existence of data in the same area in the others. Thus, we propose *seed-driven solutions*, which use one source (the one richer in the number of locations) as a *seed* to query the rest of the sources. We use the term *Multi-Source Seed-Driven* or *MSSD*, given that we simultaneously query multiple sources with points from a *seed*. The proposed algorithms can be classified as *seed-oriented* or *source-oriented*. The seed-oriented algorithms are based on the heuristic that

the distribution of the sources' locations is similar, so we can estimate the query parameters in the seed and use them to query the source. The source-oriented algorithms have some initial knowledge from the seed, but they try to learn the parameters while performing live API calls on the sources.

3.2.1 Seed-oriented algorithms

These algorithms use the seed points (p) in the API requests, and they either use a default fixed radius or calculate a suitable radius using the seed. We propose three seed-oriented algorithms: *Multi-Source Seed-Driven Fixed* (*MSSD-F*), *Multi-Source Seed-Driven Density-based* (*MSSD-D*), and *Multi-Source Seed-Driven Nearest neighbor* (*MSSD-N*).

MSSD-F. The procedure of *MSSD-F* starts with selecting a set of sources and the richer source as *seed*. Then, for each point p in *seed*, we query the remaining sources with their corresponding APIs, using p and a fixed radius r. The retrieved locations L_p^r are unioned together for each source.

MSSD-D. Instead of using a fixed r, we try to adapt the radius based on the area's density. Having a big radius is not a good solution for nearby points since the maximal result size limits us, and we might retrieve intersecting samples, while a small radius might miss locations that could have been retrieved if we were less restrictive. Thus, we would prefer a smaller radius for dense areas and a larger radius for sparser ones. For each point p in *seed*, we count the number of points $|N|$ in $Circle(p, r)$ and adjust the radius to $r_d = \frac{r}{|N|}$. Thus, we query the sources with $\langle p, r_d \rangle$.

MSSD-N. The radius of *MSSD-N* is fixed based on the nearest neighbor location for each point in the *seed*. For each point p in the *seed*, we find the point q in the *seed* such that $|p - q|$ is minimal. Then, we set $r_n = |p - q|$ and we query the rest of the sources with $\langle p, r_n \rangle$.

3.2.2 Source-oriented algorithms

In the seed-oriented solutions, we calculated the radius in the seed and then queried the sources with it, but this pre-calculated radius might not always be well-suited for the source. One can expect some similarity of the location distribution between sources, but this is not always the case, given that each source has a different scope, locations, coverage, and usage. Hence, a better solution would be finding a better radius by querying the sources. Note that the calculations that *MSSD-F*, *MSSD-D*, and *MSSD-N* perform are on the *seed* and they do not use new API requests; they are performed on the initial

points extracted from the *seed*. If we query the sources to learn a good radius, we would need extra API calls.
For the base foundation of the source-oriented algorithms, we will use the assumption that if the source contains $|L_p^r|$ locations in $Circle(p,r)$ than are less than the maximal result size M_S, then querying with $\langle p,r\rangle$ will retrieve all L_p^r. For example, if Foursquare has 10 locations in $Circle(\langle 55.8,7.9\rangle, 500\ meters)$ and the maximal result size is 20, then the API request with $\langle\langle 55.8,7.9\rangle, 500\ meters\rangle$ will return all the ten underlying locations. According to the documentations of the APIs, this assumption mostly holds in practice.

Assumption 1.1. *For each source S in $\{S_1,S_2,...,S_k\}$, if $Circle(p,r)$ contains $L_p^r(S)$ locations such that $|L_p^r(S)| \leq M_S$, then $API(p,r)$ will retrieve $L_p^r = L_p^r(S)$.* (Reproduced from [51])

Given this assumption, our goal is to find a big enough radius to cover the maximal results size number of locations. Let us illustrate with an example:

Example 1.1
Let us suppose that the maximal result size in Twitter is 3, so $M_S = 3$. We show an example of the API queries in Fig. 1.4, where in the left we are using a big radius, and we get 3 tweets for both API requests, and in the right, we use a small radius and get 3 tweets for the first request and only one for the second request. Unfortunately, we have a union of 4 tweets in both cases because there is an overlap in the results in the first case. Rather than having these pre-defined radii, we should query the sources until we reach a good radius size for each case, and ideally, obtain 6 tweets from both API requests.

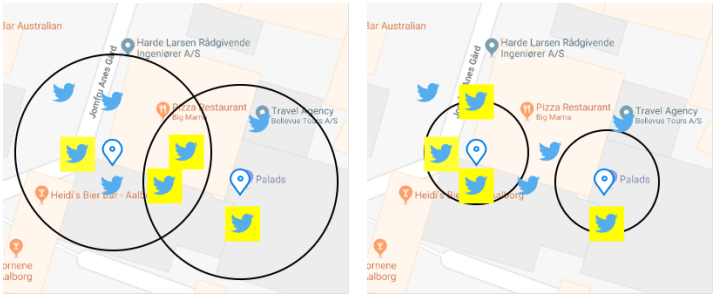


Fig. 1.4: Radius adjustment (Reproduced from [51])

Given Assumption 1.1, we can now introduce the following theorem, that provides a stopping condition when searching for a good radius.

Theorem 1.1. *Let $< p, r >$ be a pair of point and radius such that $API(p, r) = L_p^r$ where $|L_p^r| < M_S$. Then, for all r' such that $r' < r$, $L_p^{r'} \subseteq L_p^r$.* (Reproduced from [51])

The proof of the theorem is relatively intuitive: if the number of locations in a $Circle(p, r)$ are less than the maximal result size M_S, querying with a smaller radius r' will not yield any new locations. Thus, we should stop at r. The proof is formally detailed in [51]. Given this finding, we can now propose two source-oriented algorithms: *Multi-Source Seed-Driven Recursive* (*MSSD-R*), and *Multi-Source Seed-Driven Optimal* (*MSSD**).

MSSD-R. This seed-driven algorithm uses each point p of the *seed* but fixes the radius by recursively querying the source. It starts with a big radius, and if the number of the locations retrieved is equal to the maximal result size, it reduces the radius with an α coefficient and queries again. This procedure recursively continues until the number of the locations retrieved are less than the maximal result size, which means that we reached the stopping condition of Theorem 1.1.

MSSD*. The previous *MSSD-R* algorithm queries the area for each point without using redundant requests (Theorem 1.1). However, some points of the *seed* might be very near each other, but the same corresponding area in the source might be sparse. Even though we will spend only one request per point, because the stopping conditions will be reached at the first try, we could have minimized the request but clustering the nearby points and querying only once. If the area turns out to be dense, we can then split the cluster into mini-clusters and query again with their centroids. Moreover, we can improve our point selection by including the retrieved points in the current cluster; thus, when we split the cluster, we re-direct the attention to the source's dense areas.
We propose *MSSD**, a recursive seed-driven algorithm that further optimizes the data extraction problem by minimizing the number of requests. *MSSD** has advantages over the previous *MSSD* algorithms because (i) it maximizes the number of locations retrieved by adaption the radius according to the source distribution, similar to *MSSD-R* (ii) it minimizes the requests by clustering the initial seed points and performing only one request per cluster (iii) maximizes further the number of locations retrieved by changing the query point p based on the distribution of the source.
The algorithm for *MSSD** is formalized in Alg. 1.1. We perform some initial default API queries to get an initial set of points. Then, we choose the richest source as *seed* in line 4. We use DBSCAN clustering [107] to cluster the points of the *seed* in line 5. For each center c in cluster C, we call RadRecursive* (Alg. 1.2), where we query with c and an initial big radius r. If the number of the

Algorithm 1.1 *MSSD** algorithm (Reproduced from [51])

Input: A set of sources $\{S_1, S_2, ...S_n\}$, radius r
Output: $\{L^*_{S_1}, L^*_{S_2}, ...L^*_{S_k}\}$
1: **for each** S in $\{S_1, S_2, ...S_k\}$ - S_{seed} **do**
2: $L_S \leftarrow L_I = \bigcup_{i=1}^{k} L_i$ →/* Initialize each L_S with L_I*/
3: **end for**
4: Let S_{seed} be the source with the most points in $\{S_1, S_2, ...S_k\}$, L_{seed} its locations and P the distinct points in L_{seed}
5: $\{C\} \leftarrow DBSCAN(P), \epsilon, m)$
6: **for each** S **do**
7: **for each** $< c, C >$ **do**
8: $L^r_p \leftarrow \emptyset$
9: $L^r_p \leftarrow$ RadRecursive*$(r, \alpha, < c, C_c >, S, \epsilon, m, L^r_p)$
10: $L_S \leftarrow L_S \cup L^r_p$
11: **end for**
12: **end for**
return $\{L^*_{S_1}, L^*_{S_2}, ...L^*_{S_k}\}$

retrieved locations $|L^r_p|$ is less than the maximal result size M_S, we stop and continue with the next cluster. Otherwise, we union the points of L^r_p with the current cluster, we perform DBSCAN again with more a smaller ϵ and m, and for each new cluster, we call RadRecursive* again.

Algorithm 1.2 RadRecursive* (Reproduced from [51])

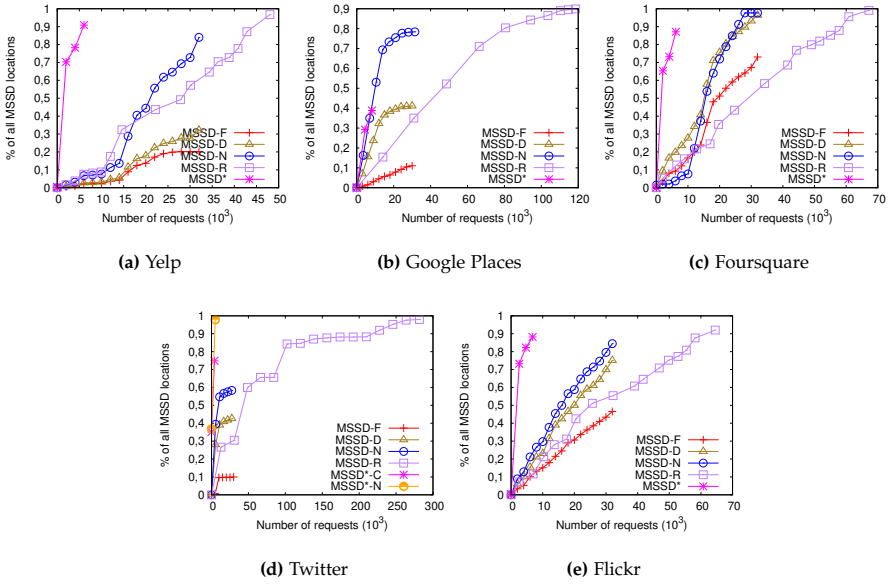
Input: r_r, α, $< p, C_p >$, S, ϵ, m, L^r_p
Output: L^r_p
1: $R \leftarrow API(p, r_r, S)$ →/* Query S with r_r*/
2: $L^r_p \leftarrow L^r_p \bigcup R$
3: **if** $|R| < M_S$ **then**
4: **return** L^r_p →/* The area is not dense*/
5: **else**
6: $\{C\}' \leftarrow DBSCAN(C_p \bigcup R, \frac{\epsilon}{\alpha}, \frac{m}{\alpha})$ →/* DBSCAN on the union of C_p and R with new parameters*/
7: **for each** $< c', C' >$ **do**
8: RadRecursive*$(\frac{r_r}{\alpha}, \alpha, < c', C'_c >, S, \frac{\epsilon}{\alpha}, \frac{m}{\alpha}, L^r_p)$
9: **end for**
10: **end if**

3.3 *MSSD* experimental results

We experimented on six location-based sources: Krak, Google Places, Yelp, Foursquare, Flickr, and Twitter. First, we queried with some default API requests to gather the initial seed points. Krak was the richer source, so it was used as a *seed* for the rest of the sources. We applied *MSSD-F*, *MSSD-D*, *MSSD-N*, *MSSD-R*, and *MSSD** on the rest of the sources (Google Places, Yelp, Foursquare, Flickr, and Twitter) using Krak points as *seed*, and the results are presented in Fig. 1.5. The number of requests is in the x-axis and percentage of locations (number of locations over the total locations retrieved by all algorithms) in the y-axis. This experiment's preferable outcome is to have as few as possible requests with as many as possible locations; thus, in the figures' left-top corner. Notably, this position is usually occupied by *MSSD**. *MSSD-R* reaches the highest percentage of locations but at a higher cost in terms of the number of requests. Moreover, *MSSD-D* and *MSSD-N* sometimes show a better trade-off between the number of locations and number of requests compared to *MSSD-R*. From the seed-oriented algorithms, *MSSD-N* retrieves the highest percentage of locations in comparison to *MSSD-F* and *MSSD-D*. Overall, *MSSD** yields 90% of the locations with only 25% of the requests of *MSSD-F*, *MSSD-N*, and *MSSD-D*. In contrast to *MSSD-R*, *MSSD** needs only 12%-15% of *MSSD-R* requests in Foursquare, Yelp and Flick, 8.5% in Google Places and 2.7% in Twitter. Hence, *MSSD* proves to be the best solution for geo-social extraction; on average, only 16% of the requests can obtain around 95% of the locations*.

Another important experiment is changing the *seed*. We tried using each source as a seed on the rest of the sources. The percentage of locations retrieved by each algorithm vs the number of requests are shown in [106], where *MSSD** again yields the best trade-off, sometimes even performing better than *MSSD-R*. For example, with Yelp as *seed* in Flickr, *MSSD** retrieves twice the locations of *MSSD-R* with 30% of the *MSSD-R* requests. We can explain this phenomenon by the fact that *MSSD* adapts the API parameters, both p and r, based on the source distribution; thus, even when the seed is not rich, it is still able to learn and outperform the other versions*.

In addition, we compared our best version of *MSSD*, *MSSD** to the baselines: *Snowball* [15, 21] and *Self-seed* [1]. *Snowball* outperforms *MSSD** in Twitter because *Snowball* uses API requests with user accounts (Fig. 1.6), which have twice the maximal result size compared to the location-based queries and do not have a limited historical access. In Flickr, *MSSD** retrieves 14 and 3 times more locations than *Snowball* and *Self-seed*, respectively (Fig. 1.6). *Snowball* is not applicable to Yelp, Foursquare and Google Places because of the lack of users. Thus, we compare *MSSD** only to *Self-seed* and *MSSD** yields 9, 5.5, and 3.5 times more locations, respectively (Fig. 1.6). A drawback of *Self-seed* is that *after a specific number of requests (experimentally 500), Self-Seed converges*



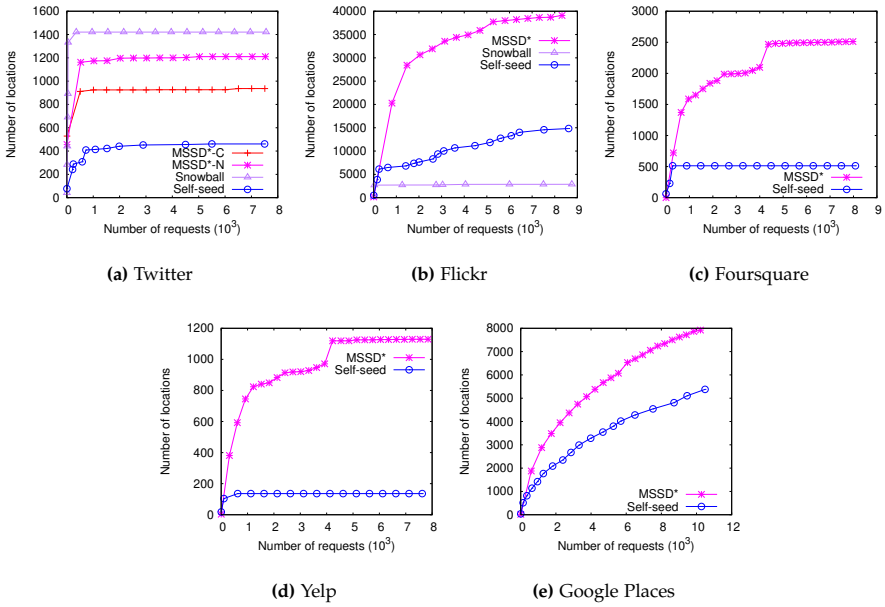
(a) Yelp **(b)** Google Places **(c)** Foursquare

(d) Twitter **(e)** Flickr

Fig. 1.5: Number of requests versus locations for different *MSSD* algorithms with Krak as seed (Reproduced from [51])

to a dead end, while the seed-driven algorithms, as multi-source solutions, are able to continue discovering new dense areas.

Discussion A multi-source solution where we use one source as *seed* for the rest of the sources showed to be a better solution than querying the source within itself, avoiding converging in a dead-end, like *Self-Seed* [1]. *MSSD** demonstrated the best trade-off between the number of requests and the number of locations, on average, only 16% of the requests of *MSSD-R* for 90% of the locations. Overall, all the algorithms improve the default API querying, 11.1 times more for the seed-oriented solution (*MSSD-F*, *MSSD-D* and *MSSD-N*) and 14.3 times for the source-oriented ones (*MSSD-R* and *MSSD**). All these data originating from different sources provide a better, holistic view of the locations in a specific area. However, this extracted data needs to be integrated and unified. In the following section, we will address the problem of spatial entity linkage, which aims to find which records (originating from the same or different sources) belong to the same physical spatial entity.



(a) Twitter **(b)** Flickr **(c)** Foursquare

(d) Yelp **(e)** Google Places

Fig. 1.6: Number of request versus number of locations for *MSSD**, *Snowball* and *Self-seed* (Reproduced from [51])

4 Multi-source spatial entity linkage

This section addresses the problem of spatial entity linkage across different sources. We first start with the problem definition; we give an overall view of the algorithms used for spatial entity linkage in this thesis; we continue with our novel spatial blocking algorithm *QuadFlex*; then, we propose three threshold-based and one unsupervised skyline-based algorithms, we introduce our main experimental results, and finally, we show the main functionalities of `skyex`, an R package for entity linkage. This section is based mostly on paper B, C, and E [52–54].

4.1 The problem of spatial entity linkage

The location-based entities extracted from geo-social sources are identified within the source with an id, but this is not usually sufficient to identify each entity uniquely. The same entity might appear more than once within the same source, with different ids, and sometimes even different attributes. This problem becomes even more challenging when including different sources. Let us first define a spatial entity as follows:

Definition 1.2. *A spatial entity s is an entity originating from a location-based source I, located in a geographical point p and accompanied by a set of attributes $A = \{a_i\}$.* (Reproduced from [52, 53])

To determine which records of these spatial entities belong to the same physical entity, we need to compare them with each other and finally come to a decision. We formalize this problem as follows:

Problem definition: Given a set of spatial entities S originating from multiple sources, the spatial entity linkage problem aims to find those pairs of spatial entities $\langle s_i, s_j \rangle$ that refer to the same physical spatial entity. (Reproduced from [52, 53, 55])

4.2 Overall solutions

In this section, we will describe the overall solutions that this thesis proposes to solve the problem of spatial entity linkage. The overall workflow is presented in Fig. 1.7 and covers several novel algorithms, *QuadFlex* [52, 53], *SkyRank* [53], *SkyEx* [52], *SkyEx-F* [53], *SkyEx-FES* [53], and *SkyEx-D* [53], and *SkyEx-T* [55] (Section 5).
We start with a set S of spatial entities, originating from single or multiple sources. We apply *QuadFlex*, a spatial blocking technique to group together spatial entities, which need to be pairwise compared. We can either compare the pairs using traditional similarity metrics, or use state-of-the-art meta-similarity features specifically tailored for spatial entities, which are calculated by *LGM-X* [55]. The solutions proposed in [52, 53] use pairwise comparisons of the name, address and semantic similarity. The skyline-based trained version proposed in [55] uses *LGM-X* features. The pairs are ranked on skylines using *SkyRank* [53] using either simple Pareto functions (*SkyEx* [52], *SkyEx-F* [53], *SkyEx-FES* [53], and *SkyEx-D* [53]), or a trained preference function (*SkyEx-T* [55]).
SkyEx-T is trained on a tiny training set to learn a preference function p and a cut-off c_t, which will later be applied to separate the classes. In contrast, *SkyEx* [52], *SkyEx-F* [53], and *SkyEx-FES* [53] are threshold-based, meaning that the user inputs the parameter k of the level of skylines in order to separate the classes. *SkyEx-F* and *SkyEx-FES* control the selection of the threshold by checking on the F-measure of each selection, but *SkyEx-FES* provides an early-stop condition to limit the search and stop earlier. *SkyEx-D* is an unsupervised density-based algorithm that does not need any parameters and still can estimate k to separate the classes. All the proposed solutions share a common inspiration for using flexible preference functions without weights and can provide very good model accuracy, high model explainability, and high robustness in deployment.

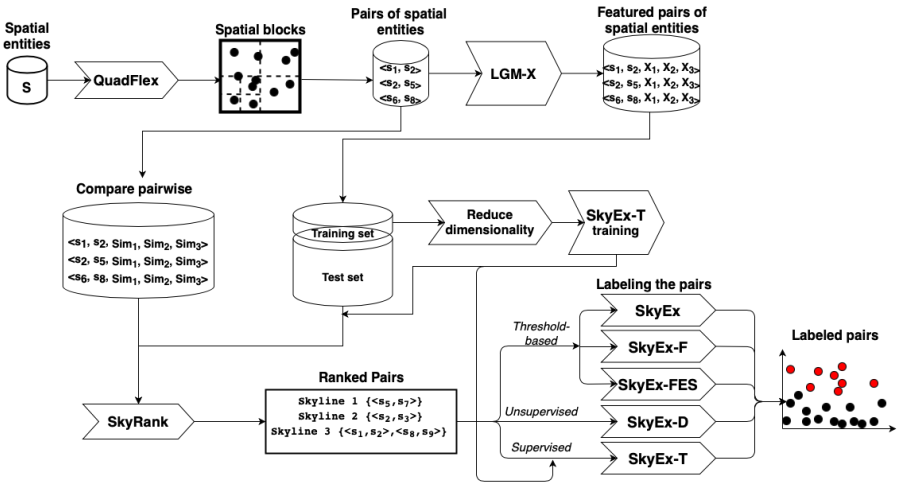


Fig. 1.7: Overall spatial entity linkage solutions

4.3 Spatial blocking with *QuadFlex*

In order to avoid comparing all spatial entities with each other, we propose *QuadFlex*, a spatial blocking algorithm based on a quadtree [108], but with the following modifications: (i) instead of the capacity, we use the density of a node as a trigger to split the node into four children, (ii) we use the diagonal of the node as a maximal spatial distance between the points to avoid comparing distant spatial entities, (iii) we allow the intersection of the nodes not to miss relevant comparisons. The algorithm is formalized in Alg. 1.3.

We first create a bounding box that contains all the entities in S and define the diagonal m and the density d for *QuadFlex* in line 1. Then, we have to add the spatial entities to *QuadFlex* in line 3. In order to do so, we call Method *insert(s)* (lines 5-16). We check if the current node has children (line 5), then we find in which of the children s can be assigned using Method *getIndex(s)* (lines 18-31). Note that Method *getIndex(s)* returns a set of indexes because s can be assigned to more than one child. If the restrictions about the d or m apply, then the node will split into four new children (line 11-17), and we re-assign s. In the end, we return the leaves of the *QuadFlex*, which are our spatial blocks.

4.4 Threshold-based skyline algorithms

After obtaining the spatial blocks, we will compare the spatial entities in each block. We compare the following attributes: name, address, and category.

Algorithm 1.3 QuadFlex algorithm (Reproduced from [52, 53])

Input: A set of entities $S = \{s_i\}$, diagonal m, density d
Output: The leaves QuadFlex Q *Q.leaves()* ;

Create $Q(m, d)$ where Q has the dimensions of the bounding box of S
for each s in S **do**
    Q.insert(s) *// Insert s into the QuadFlex*
end for
return *Q.leaves()*

Method *insert (s)*
if *this.children* $\neq \emptyset$ **then**
    Indexes $\leftarrow$ *getIndex(s)* *// Find where s belongs*
    for each i in *Indexes* **do**
        this.child[i].insert(s) *// Insert s to the children it belongs*
    end for
end if
if *this.diagonal* $> m$ **or** *this.density* $> d$ **then**
    Split the current object *this* into 4 children
end if
Indexes $\leftarrow$ *getIndex(s)*
for each i in *Indexes* **do**
    this.child[i].insert(s)
end for
return

Method *getIndex (s)*
Let *vertical-left* and *vertical-right* be the lines that pass at 0.25 and 0.75 of the width of *this*, respectively
Let *horizontal-up* and *horizontal-down* be the lines that pass at 0.25 and 0.75 of the height of *this*, respectively
if s is left of *vertical-right* and above *horizontal-down* **then**
    Indexes.add(1) *// s fits in child[1]*
end if
if s is right of *vertical-left* and above *horizontal-down* **then**
    Indexes.add(2) *// s fits in child[2]*
end if
if s is left of *vertical-right* and below *horizontal-up* **then**
    Indexes.add(3) *// s fits in child[3]*
end if
if s is right of *vertical-left* and below *horizontal-up* **then**
    Indexes.add(4) *// s fits in child[4]*
end if
return *Indexes*

The name and the address and compared using string similarities, while for the category, we use Wu&Palmer similarity [109] with Wordnet [110]. After having all three similarities (*SimName*, *SimAddress*, and *SimSemantic*), we have to decide which pairs of spatial entities belong to the same physical entity and label them as 1, or as 0, otherwise.
Instead of creating a scoring function or using machine learning, we rank the pairs, and we select a cut-off k to separate the ranked pairs into classes. Let us first start with the central concept, the Pareto optimality [111]. *A solution (x, y) is Pareto optimal when no other solution can increase x without decreasing y.* Using this concept, we can start by finding those pairs that dominate the rest, meaning that there are no other solutions that can improve in terms of one of the similarities without decreasing the other.

Definition 1.3. *A skyline of level k, $Skyline(k)$, is the collection of pairs $\langle s_i, s_j \rangle$ ranked in the k^{th} position such that for each pair $\langle s_i, s_j \rangle \in Skyline(k)$ and for each pair $\langle s'_i, s'_j \rangle \in Skyline(k')$ where $k' > k$, $\langle s_i, s_j \rangle$ dominates $\langle s'_i, s'_j \rangle$ ($\langle s_i, s_j \rangle \succ \langle s'_i, s'_j \rangle$).* (Reproduced from [55]).

The definition in [52, 53] express this dominance in terms of a *utility*, denoted as $u(\langle s_i, s_j \rangle)$, whose value indicates the possibility of the pair $\langle s_i, s_j \rangle$ to be a match or not. A higher utility means a higher chance for the pair to be the same physical entity. Both definitions refer to the same rationale; a pair ranked higher is more likely to be a match; thus, it is more preferred than another pair ranked lower. The algorithm for ranking the pairs is formalized in Alg. 1.4. We start with all the pairs, find the first skyline *Skyline(1)* that dominate the rest, remove *Skyline(1)* from the initial set P and put it to P_k. Then, we find the next skyline and continue until there are no more unranked pairs. The output of *SkyRank* is pairs of spatial entities and their skyline level k, $P_k = \{\langle s_i, s_j \rangle, k\}$.

Algorithm 1.4 Skyline Ranking (SkyRank) (Reproduced from [53])

Input: A set of pairs $P = \{\langle s_i, s_j \rangle\}$
Output: A set of pairs and their skyline $P_k = \{\langle s_i, s_j \rangle, k\}$;

```
P_k ← ∅
while |P_k| < |P| do
    Filter Skyline(k) = {⟨s_i, s_j⟩} | ∀⟨s', s''⟩ ∈ P − {⟨s_i, s_j⟩} , u(⟨s_i, s_j⟩) > u⟨s', s''⟩}   // Find the Skyline
    Add Skyline(k) to P_k   // Move the skyline to P_k
    P = P − Skyline(k)
return P_k
```

SkyRank is the base algorithm for the *SkyEx-** family of algorithms, which,

besides ranking the pairs, also decide the cut-off k that best separates the classes. The threshold-based versions *SkyEx* [52], *SkyEx-F* [53], and *SkyEx-FES* [53] require k as an input.

SkyEx. *SkyEx* [52] needs k as an input from the user. Given that the user might have some domain knowledge, he can set a k value. Then, all the pairs that are ranked from [1,k] will be labeled as positive and the rest as negative. As a result, given this cut-off, *SkyRank* will continue until the k^{th} rank instead of continuing for all the pairs in P.

SkyEx-F. Different cut-offs yield different precision, recall and F-measure. *SkyEx-F* [53] is a threshold-based version as well, but it experimentally tries different cut-offs and fixes the cut-off k such that the F-measure is maximized.

SkyEx-FES. *SkyEx-F* tries exhaustively every cut-off until finding the best one. *SkyEx-FES* [53] is based on a theoretical guarantee, proven in [53], which guarantees that moving from the first skyline to the rest, the F-measure increases, reaches the maximal value, and then decreases without the possibility to improve later. Given this finding, we can search until the local-maximal value of F-measure and discard the rest of the ranking.

SkyEx, *SkyEx-F*, *SkyEx-FES* are based on the same rationale of a user-based threshold. *SkyEx* relies on the knowledge of the user to find a good threshold, while *SkyEx-F* and *SkyEx-FES* set a practical example of how to fix the threshold. In the next sections, we will present an unsupervised and a lightly trained version that is able to estimate a near-optimal k.

4.5 Unsupervised skyline-based algorithm

In this section, we will introduce an unsupervised skyline-based algorithm *SkyEx-D* that can separate the classes by considering the distance between the skylines. The pairs that are likely to be a match are usually in the first skylines. In contrast to clustering, where similar points will have a small intra-cluster distance, the points in a skyline do not necessarily exhibit this behaviour; they simply share the same utility. However, they tend to distance themselves from the rest of the skylines that contain pairs which are not likely to be a match. Thus, we measure the distance between the pairs starting from the first skyline: this distance is small initially, increases where the skylines best separate the classes and starts decreasing when we enter the skylines that contain negative pairs.

We denote the distance between the classes as $\mu_d(k) = \frac{\sum d(p^k, p^{-k})}{|P_k|}$, where P_k is the set of pairs from the 1^{st} to the k^{th} skyline, p^k is a pair in P^k, p^{-k} is a pair

in $P - P^k$, and $d(p^k, p^{-k})$ is the distance between p^k and p^{-k} [53].
We monitor the evolution of $\mu_d(k)$ starting from the first skyline using its derivative $\mu'_d(k) = \frac{\partial}{\partial k} \approx \frac{\mu_d(k+1) - \mu_d(k)}{1}$. When $\mu'_d(k) > 0$, $\mu_d(k)$ has a positive slope, meaning that the distance is increasing. When $\mu'_d(k) < 0$ for the first time, then we are entering the skylines with negative pairs, and the longer we continue, the more we might loose in precision. Therefore, we stop the first time that $\mu'_d(k) < 0$. However, $\mu'_d(k)$ can be sensitive to small fluctuations and we might cut-off too early, so we smooth $\mu'_d(k)$ with the Gaussian function ($\frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2 / 2\sigma^2}$) and then we fix the values k_d where $\mu'_d(k) < 0$ for the first time.
SkyEx-D algorithm is formalized in Alg. 1.5. First, Alg. 1.4 ranks the pairs and assigns them skylines, and in the meantime, it calculates $\mu'_d(k)$ when moving from one skyline to the next in line 7. Then, for each skyline, we note skyline of level k_d when the smoothened $\mu'_d(k) < 0$ for the first time (the lines 8-14). Finally, we label all the pairs in skylines $[1, k_d]$ as positives and the rest as negatives in lines 16-17.

Algorithm 1.5 SkyEx-D (Reproduced from [53])

Input: A set of pairs $P = \{\langle s_i, s_j \rangle\}$
Output: A set of positive pairs P^+ and a set of negative pairs P^- ;
1: $P_k \leftarrow \emptyset$
Lines 2-6 as Algorithm B.2...
7: Calculate $\mu'_d(k)$ in each k
8: **while** $k < k_{last}$ **do**
9: **if** $smooth(\mu'_d(k)) < 0$ **then**
10: $k_d \leftarrow k$
11: **break**
12: **else**
13: $k \leftarrow k + 1$
14: **end if**
15:
16: $P^+ \leftarrow \bigcup_{k=1}^{k_d} Skyline(k)$
17: $P^- \leftarrow P_k - P^+$
return P^+, P^-

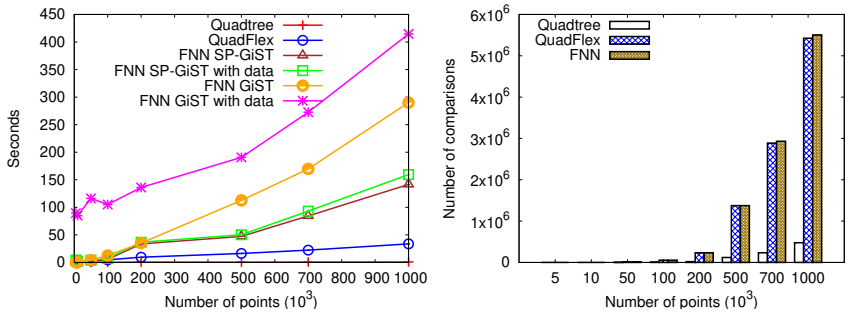
4.6 Experiments with threshold-based and unsupervised SkyEx-* algorithms

This section will summarize the main experimental results from [52, 53]. The dataset is extracted in [51]. It contains 75,541 spatial entities from Google

Places (GP), Krak, Foursquare (FSQ), and Yelp. For the ground truth, we automatically labeled the pairs as positive if the phone number or the website is the same, and as negative, otherwise. Additionally, we manually labeled 1500 random pairs. We refer to the manually labeled pairs as D_{sample}, and to the automatically labeled pairs as D_{full}.

4.6.1 Performance of QuadFlex

We compare *QuadFlex* (Java implementation) with the quadtree (Java implementation) and Fixed Radius Nearest Neighbors algorithm [112] FNN (PostgreSQL[1] using spatial indexes: GiST[2] and SP-GiST[3]). We simulate up to 1,000,000 random points besides our dataset to test the approaches using different densities. The results are shown in Fig. 1.8. The FNN versions with data add the time to move the pairs from PostgreSQL to Java to the execution time, while for the FNN versions without data, we only measure the execution time in PostgreSQL. We can distinguish that *QuadFlex* is the second-best after the quadtree, but note that the quadtree misses around 90% of the comparisons (Fig. 1.8b). FNN SP-GiST performs slightly better than *QuadFlex* initially, but later, when the density increases, *QuadFlex* performs around eight times faster than FNN GiST and 3 times faster than FNN SP-GiST. FNN with SP-GiST index is faster than FNN GiST for all the densities. We also tried not having any index (not in Fig. 1.8 because it would have dwarfed the other curves), and it turned out that it is up to 848 times slower than FNN Gist with data, and up to 368,095 times slower than *QuadFlex*. To sum up, *QuadFlex performs 99.99% of the comparisons of FNN with a performance comparable to a quadtree*.



(a) Execution time **(b)** Number of comparisons

Fig. 1.8: Comparing quadtree, QuadFlex and FNN (Reproduced from [52, 53])

[1]https://www.postgresql.org
[2]https://www.postgresql.org/docs/current/gist.html
[3]https://www.postgresql.org/docs/current/spgist.html

4.6.2 Evaluation of SkyEx-* algorithms

In this section, we compare the results from *SkyEx*, *SkyEx-F*, *SkyEx-FES*, and *SkyEx-D* to the similar work on spatial entity linkage [80, 91, 104], to supervised learning algorithms and to clustering [unsupervised] techniques. Berjawi et al. [104] compare the pairs considering their distance and the textual attributes using Levenshtein similarity, and finally, the similarities are all added together. The pairs are considered a match if the score is higher than 0.75. We use 0.75 but also try different thresholds are reported the best one (the versions with the suffix *-Flex*). We also denote the two different version proposed in [104] as: V1 for name + address + geographic coordinates, V2 for name + geographic coordinates. Morana et al. [91] compare only entities that share tokens in the name or the same category. The pairs are compared considering their spatial distance, name and address (Levenshtein), and category (Resnik similarity using Wordnet). The similarities are added to a scoring function with weights: $\frac{1}{3}$ for address, and phone and $\frac{2}{3}$ for the name, category, and spatial similarity. Finally, a user can choose from the top k most similar entities. In Karam et al. [80] the pairs at most 5 m apart are compared considering the name (Levenshtein distance), the geographic similarity, and the keywords (semantic similarity). The belief theory [105] decides which pairs will be labeled as positive. The results are shown in Table 1.1. Even

	D_{full}			D_{sample}		
Approach	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Berjawi et al.(V1) [104]	**0.93**	0.26	0.41	**1.00**	0.27	0.43
Berjawi et al.(V1) [104]-*Flex*	0.87	0.50	0.63	0.79	0.42	0.55
Berjawi et al.(V2) [104]	0.73	0.56	0.63	0.97	0.60	0.74
Berjawi et al.(V2) [104]-*Flex*	0.73	0.56	0.63	0.82	0.76	0.79
Morana et al. [91]	0.39	0.60	0.47	0.33	0.60	0.43
Karam et al. [80]	0.23	**0.73**	0.35	0.54	0.68	0.60
QuadSky with *SkyEx/SkyEx-F/SkyEx-FES*	0.87	0.60	**0.72**	0.87	**0.82**	**0.85**
QuadSky with *SkyEx-D*	0.85	0.62	0.71	0.87	**0.82**	**0.85**

Table 1.1: Comparison with the baselines (Reproduced from [52, 53])

though some of the methods might stand out in terms of precision (Berjawi et al.(V1) [104]) or recall (Karam et al. [80]), *QuadSky has the best F-measure in both datasets*. Moreover, *QuadSky* offers a more balanced model in precision and recall combined. Interestingly, *SkyEx-D, even though fully unsupervised, can find a cut-off that yields an F-measure of only 0.01 smaller than the F-measure of SkyEx/SkyEx-F/SkyEx-FES*.

We also experimented with replacing the skyline-based algorithm for the labeling of the pairs with some common supervised learning algorithms that have been using in entity linkage [56, 83, 102, 113, 114]: Logistic Regression [115], Support Vector Machines [116], Decision Trees [117], and Naive

Bayes [118]. We trained on 75% of the data and tested on the remaining 25% with 4-fold cross validation on the whole dataset (D_{full}-D_{full}), and on the manually labeled pairs (D_{sample}-D_{sample}). Additionally, we trained on D_{sample} and testing on D_{full} (D_{sample}-D_{full}). The results are shown in Table 1.2. For D_{full}-D_{full} and D_{sample}-D_{sample}, our approaches differ from the best method with 0.02-0.03 in F-measure. *However, in a more realistic scenario where one would have to manually label some data and test them on a larger dataset (D_{sample}-D_{full}), the SkyEx-* algorithms outperform the supervised techniques*.
Additionally, we tried compared *SkyEx-D* to clustering techniques: distance-based clustering (k-means [119] and k-medoids [120]), hierarchial clustering [121] (agglomerative), and density-based clustering (DBSCAN [107]. The results are presented in Table 1.3. Most of the clustering techniques yield a very high recall but at the cost of low precision and thus a low F-measure. *In both datasets, SkyEx-D outperforms all the clustering techniques by far higher F-measure*.

	D_{full} -D_{full}			D_{sample}-D_{sample}			D_{sample}-D_{full}		
Method	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Logistic regression	0.83	0.70	**0.76**	0.80	0.83	0.81	0.70	0.72	0.71
SVM	**0.88**	0.67	**0.76**	0.81	0.80	0.81	0.71	0.70	0.71
Decision Trees	**0.88**	0.66	0.75	**0.93**	0.82	**0.87**	0.65	0.74	0.69
Naive Bayes	0.71	**0.77**	0.74	0.63	**0.85**	0.72	0.62	**0.77**	0.69
SkyEx/SkyEx-F/SkyEx-FES	0.80	0.69	0.74	0.87	0.82	0.84	0.80	0.69	**0.74**
SkyEx-D	0.81	0.68	0.74	0.87	0.82	0.84	**0.81**	0.68	**0.74**

Table 1.2: Comparison with supervised learning (Reproduced from [53])

	D_{full}			D_{sample}		
Method	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
K-means	0.28	0.96	0.44	0.62	0.92	0.74
K-medoids	0.28	0.96	0.44	0.62	0.92	0.74
Hierarchial	0.62	0.11	0.19	0.23	0.91	0.36
DBSCAN	0.23	**1.00**	0.37	0.26	**1.00**	0.42
SkyEx-D	**0.81**	0.68	**0.74**	**0.87**	0.82	**0.84**

Table 1.3: Comparing SkyEx-D to clustering techniques (Reproduced from [53])

4.7 `skyex` R package

In order to offer a user-friendly and easy to plug-in tool, we implemented `skyex`, an R package that can handle the entire pipeline of an entity linkage process, but more importantly, includes the novel skyline-based solutions

SkyEx-F and *SkyEx-D* [53] for pair labeling. Additionally, `skyex` supports analysis and visualization functions that help interpret the results.
The modules and the functions of `skyex` are listed in Table 1.4. We offer textual and spatial blocking in the Blocking module. The `textual_blocking` uses some typical string metrics such as Cosine, Jaccard, etc. However, sometimes simply checking for a common prefix or suffix could yield good results and, as expected, have a shorter run time. The pairwise comparison can be performed considering the textual, spatial, and semantic attributes of two entities. The text similarity uses typical similarity metrics; the spatial similarity is a normalized value using a maximal distance in meters; the semantic similarity is based on Wordnet, specifically on Path and Wu&Palmer similarities. Example scripts for blocking and pairwise comparisons are shown below:

```
#spatial blocks of 50 m
blocks<-spatial.blocking(data=entities, longitude="long",
                                    latitude = "lat", max_distance = 50)
#textual blocks with maximal Levenshtein distance of 2
blocks<- textual.blocking(data=entities, column="name",
                                     method="levenshtein", max_distance=2)
#pairwise text similarity of the address
blocks$SimAddress <- text.similarity(data = blocks,
        column1 = "address.x", column2 = "address.y", method = "jaccard")
```

The most important modules of `skyex` are the Labeling, and the Analysis and Visualization modules, because while other tools offer blocking techniques and pairwise comparison functions [122–128], `skyex` implements the novel skyline-based algorithms and functions to support a good and solid analysis. We first start by defining the preference function. We decide which similarities will be part of the Pareto function. Let us suppose we choose the name, address, and semantic similarity. We can call `skyexf` for a threshold-based or `skyexd` for the unsupervised solution.

```
#Creating the preference function
p<-high(SimName)*high(SimAddress)*high(SimSemantic)
#using SkyEx-F
f.obj <- skyexf(data=blocks, p = p, label = "Class",
                posclass = 1, negclass = 0)
#using SkyEx-D with a smoothing of 10
d.obj <- skyexd(data=blocks, p = p,
                simlist = c("SimName", "SimAddress", "SimSemantic"),
                smooth.coefficient = 10, posclass = 1, negclass = 0)
```

We can simply extract the predicted classes from `f.obj` and `d.obj`, or we can do further analysis. For example, for *SkyEx-F* we can plot the different cut-offs and the values of precision, recall and F-measure using `plot.skyexf.cutoffs` (Fig. 1.9). For *SkyEx-D*, we can see the cut-off using `plot.skyexd.cutoffs` and we can adjust the smoothing coefficient without having to re-run the algorithms by using `plot.skyexd.smooth` (Fig. 1.10).
Additionally, we can call `evaluate.skyex` to see the precision, recall and F-measure, or call `plot.pairs2D` (Fig. 1.11a), `plot.pairs3D`, and `plot.pairs.interactive.3D` (Fig. 1.11b) to see with different colors the pre-

Module	Function name	Decsription
Blocking	`textual_blocking`	Text blocks (Levenshtein, Jaccard, Cosine, Jaro-Winkler, qgram)
	`spatial_blocking`	Spatial blocks based on distance in meters
	`prefix_blocking`	Blocking based on textual prefix (nr of characters)
	`suffix_blocking`	Blocking based on textual suffix (nr of characters)
Pairwise comparison	`text_similarity`	Text similarity (Levenshtein, Jaccard, Cosine, Jaro-Winkler)
	`spatial_similarity`	Spatial similarity normalized by max. distance
	`semantic_similarity`	Semantic similarity (Path or Wu&Palmer)
Labeling	`skyexf`	Labeling the pairs with SkyEx-F
	`skyexd`	Labeling the pairs with SkyEx-D
Analysis and visualization	`plot.skyexf.cutoffs`	Plot SkyEx-F cut-offs
	`plot.skyexd.cutoffs`	Plot SkyEx-D cut-offs
	`plot.skyexd.smooth`	Plot SkyEx-D for different smoothing coefficients
	`evaluate.skyex`	Evaluate SkyEx algorithms (precision, recall, F-measure)
	`block.positives.coverage`	Check the quality of the blocking
	`plot.pairs2D`	Plot SkyEx labeled pairs 2D
	`plot.pairs3D`	Plot labeled SkyEx pairs 3D
	`plot.pairs.interactive.3D`	Interactive plots 3D for labeled SkyEx pairs

Table 1.4: Modules and functions of `skyex`

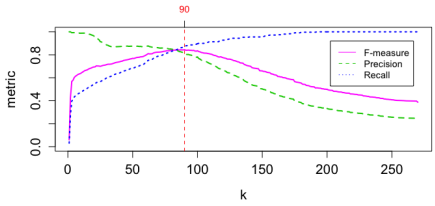
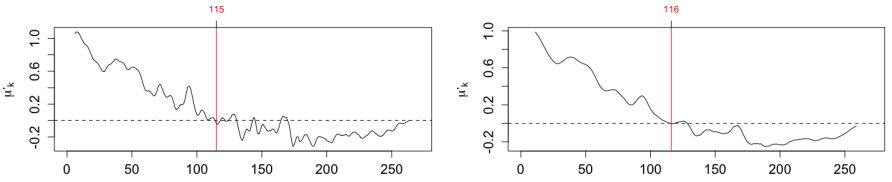


Fig. 1.9: Plotting all metrics with `plot.skyexf.cutoffs`

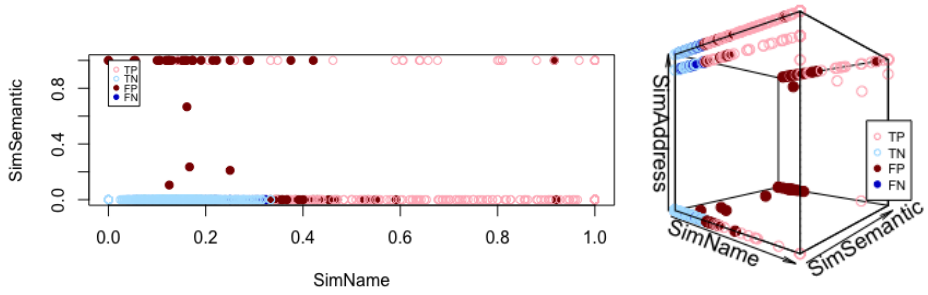
dicted and the actual classes when the data has two or three dimensions, respectively.

Discussion We addressed the problem of spatial entity linkage with an effective and flexible spatial blocking algorithm *QuadFlex*, compared the pairs pairwise syntactically and semantically, and labeled them using threshold-based and unsupervised novel skyline-based algorithms. The SkyEx-* algorithms do not use scoring functions, weights, or training sets, and despite their light configuration, they were able to outperform the baselines and the unsupervised techniques. Even though supervised techniques had a slightly higher F-measure in a traditional setting, when trained on a small set and tested on a larger one (D_{sample}-D_{full}), the SkyEx-* algorithms yielded better results. The Pareto function offers a flexible method to aggregate the similarity scores of a pair, but it lacks the expressiveness to accommodate more specific relations, such as one similarity being more important than another. Moreover, our success in D_{sample}-D_{full} points out that a small labeled set might be able to lightly train the skyline-based algorithm. We will address these two



(a) `smooth.coefficient=10` **(b)** `smooth.coefficient=20`

Fig. 1.10: Performing analysis using `plot.skyexd.cutoffs` and `plot.skyexd.smooth`



(a) 2D plotting **(b)** 3D plotting

Fig. 1.11: Visualizing the results

promising directions in the next sections.

5 Skyline-based spatial entity linkage algorithm trained on tiny data

SkyEx, *SkyEx-F*, *SkyEx-FES*, and *SkyEx-D* rank the pairs using the Pareto dominance concept so they do not prioritize any attribute similarity over others. Moreover, *SkyEx*, *SkyEx-F*, and *SkyEx-FES* still require that the user provides the cut-off value (the number of skylines k). In this section, we propose a lightly trained skyline-based algorithm *SkyEx-T* that can learn the preference function and the cut-off in a small training set. This section is based on paper D [55].

5.1 *SkyEx-T* concepts

SkyEx-T uses the state-of-the-art meta-similarity function specially tailored for spatial entities *LGM-X*, which extends *LGM-Sim* [96, 98] to accommodate the different similarity features for the address. We refer to the output of *LGM-X* as *features* and to the pairs accompanied with the *LGM-X* features as

featured pairs (see Fig. 1.7).
After having the featured pairs, we split them into non-intersecting training and test sets. We train *SkyEx-T* on the small training set and apply it on the test set. Let us start with the main concepts of *SkyEx-T*:

Definition 1.4. *A preference function* $p : P \mapsto P_k$ *is a function that takes as input a set of pairs* $P = \{\langle s_i, s_j \rangle\}$ *and outputs the partially ranked (some pairs share the same rank) list of pairs* $P_k = \{\langle s_i, s_j, k \rangle\}$ *with respect to their feature values, where* $k \in [1, m]$ *is the rank of a pair* $\langle s_i, s_j \rangle$*, and* m *is the maximal rank.* (Reproduced from [55])

The preference function is constructed using two components: *the preferred feature direction* and *the preference operators*.

Definition 1.5. *The preferred feature direction* $d() : X \mapsto \mathcal{N}$ *is a function that takes as input the list of values of a feature* X*, orders them preferring either high values* $(high())$ *or low values* $(low())$*, and outputs the rank for each feature.* (Reproduced from [55])

For the preference function to prefer a pair, it needs to know if high or low values of a feature are preferred to rank them accordingly. This concept has been indirectly addressed in [52, 53], when we identify if an attribute is *positive discriminating*, meaning that its high values usually indicate a match. However, the work in [52, 53] did not accommodate features that rather indicate a non-match. For example, a high value of Levenshtein distance is an indicator of a non-match, while a high value of Jaccard points to a match. So, we want *low(Levenshtein distance)* and *high (Jaccard)* .
These feature directions can be connected together with a *Pareto* operator or a *Priority* operator, both defined as below:

Definition 1.6. *The Pareto operator* $\triangle$ *is a binary operator connecting two preferred feature directions according to the Pareto Optimality concept.* (Reproduced from [55])

Definition 1.7. *The Priority operator* $\triangleright$ *is a binary operator connecting two preferred feature directions such that the feature direction on the left side of the operator is preferred over the one on the right side.* (Reproduced from [55])

The ranking of the pairs into skylines in [52, 53] was done by using only the Pareto operator, while *SkyEx-T* offers the possibility to prioritize some feature over others. Let us give an example:

Example 1.2
Let $\langle s_1, s_2 \rangle$ be a pair of spatial entities. X_1=0.7 and X_2=0.3 are features that expressed the similarly of the name attribute, while $X_3 = 10$ shows the

distance in meters from s_1 to s_2. We prefer high values of X_1 and X_2, so we have $high(X_1)$ and $high(X_2)$. The high distance between entities is an indicator of dissimilar spatial entities, so we prefer a low distance $low(X_3)$. Furthermore, X_2 shows to be better at detecting the class than X_1 and X_3, while we are indifferent between X_1 and X_3. Therefore, when constructing the preference function, we connect X_1 and X_3 with the Pareto operator, and we prioritize X_2. Finally, we formulate the preference function $p = high(X_2) \triangleright (high(X_1) \triangle low(X_3))$.

The full procedure of *SkyEx-T* is represented in Fig. 1.12. We start with the featured pairs produced from *LGM-X*, and we split then into a small training set and the rest as a test set. We reduce the dimensionality of the features in the training set, and the output is used to learn a preference function p. Later, the pairs in the training set are ranked according to p, and we fix the cut-off c_t that maximizes the F-measure. Finally, we rank the pairs in the test set according to the preference p, separate the classes based on c_t and return the labeled pairs.

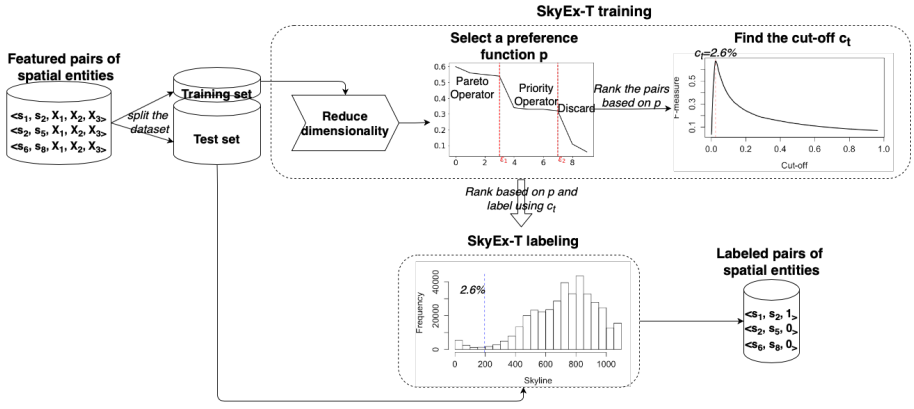


Fig. 1.12: SkyEx-T pipeline

5.2 *SkyEx-T* algorithm

LGM-X produces multiple features, which can be highly correlated. Thus, we need to reduce the dimensionality. We use *mutual information* (MI) [129] to remove the highly correlated features. $MI(x,y) = \int_x \int_y p_{x,y}(x,y) log \frac{p_{x,y}(x,y)}{p_x(x)p_y(y)}$, where $p_x(x)$ and $p_y(y)$ are the marginal probability density functions of variables x and y and $p_{x,y}(x,y)$ is the joint probability function of x and y. We

use the pairs with reduced number of features to learn a preference function that prefers the pairs that are likely to be a match over the rest.

Algorithm 1.6 SkyEx-T training (Reproduced from [55])

Input: A set of labeled pairs $P_t = \{\langle s_i, s_j, C_{ij}\rangle\}$
Output: A trained preference function p and cut-off c_t

$R_X \leftarrow \emptyset$
Calculate ρ_{X_i} for each X_i and add each $|\rho_{X_i}|$ to R_X
Order R_X in a descending order
Find ε_1 and ε_2 elbows in R_X
for each X_i that $|\rho_{X_i}| \leq \varepsilon_1$ **do**
    $p_1 = d(X_1) \triangle d(X_2) ... \triangle d(X_m)$
end for
for each X_i that $|\rho_{X_i}| > \varepsilon_1$ and $|\rho_{X_i}| \leq \varepsilon_2$ **do**
    $p_2 = d(X_m + 1) \triangle d(X_{m+2}) ... \triangle d(X_n)$
end for
$p = p_1 \triangleright p_2$
$P_k \leftarrow \emptyset$
$F \leftarrow \emptyset$
while $|P_k| < |P|$ **do**
    Find $Skyline(k) = \{\langle s_i, s_j\rangle\} \mid \forall \langle s', s''\rangle \in P_t - \{\langle s_i, s_j\rangle\}$, $\langle s_i, s_j\rangle \succ \langle s', s''\rangle\}$
    Remove $Skyline(k)$ from P and add it to P_k
    Label P_k as positive
    Calculate $F1(k)$ and add $F1(k)$ to F
    $P_t = P_t - Skyline(k)$

Find k_l such that $F1(k_l) = max(F1(k))\ \forall k \in \{1, |F|\}$
$c_t = \frac{\sum_{i=1}^{k_l} Skyline_i}{\sum_{i=1}^{max(k)} Skyline_i}$
return p, c_t

For the selection of a preference function, we first evaluate the correlation of each feature to the class. The preference function should contain features whose increase or decrease in value affects the value of class C. Thus, we use Spearman's correlation $\rho_{X_i} = \frac{cov(X_i, C)}{\sigma_{X_i}\sigma_C}$, where σ_{X_i} and σ_C account for the standard deviations of X_i and C, respectively, while $cov(X_i, C)$ is the covariance of X_i and C. Judging from the values of ρ_{X_i}, we can now decide whether a feature will be prioritized or not. We plot the descending absolute values of ρ_{X_i} and find two elbows ε_1 and ε_2. The features with $|\rho_{X_i}| \leq \varepsilon_1$ (X_{ε_1}. X_{ε_1}) have a stronger monotonic relationship to C than the features with $|\rho_{X_i}| \leq \varepsilon_2$ and $\rho_{X_i} > \varepsilon_1$ (X_{ε_2}), so we will prioritize X_{ε_1} over X_{ε_2}. In the meantime,

we will use the Pareto operator amongst X_{ε_1} and amongst X_{ε_2} to treat them equally.
After having the preference function p, we rank the pairs in the training set according to p. We find k_t number of skylines that yield the maximal F-measure. Given that the training and the test set originate from the same population, they share similar probability density distributions. We prove in [55] that the distribution of the ranked pairs using the same preference function p also share similar probability density distributions (Theorem D.1 in [55]). Consequently, if a cut-off in the training set is highly preferred, it will also be preferred in the test set [130] (Theorem D.2 in [55]). Using the data ratio c_t of the pairs belonging to the skylines up to the k_t^{th} level over the total pairs will still yield a near-optimal cut-off for the test set (Lemma D.1 in [55]). Therefore, we fix c_t as the optimal cut-off and use it to separate the classes in the test set.
The above procedure is formalized in Alg. 1.6. The preference training is handled by the lines 1-11. We calculate for each feature the Spearman's correlation to the class and add it to R_X. We order R_X in line 3 and find the two elbows ε_1 and ε_2 in line 4. Then, we construct the preference function p by using the Pareto operator for the X_{ε_1} features (lines 5-7) and X_{ε_2} features (lines 8-10), and finally, we use the Priority operator for X_{ε_1} over X_{ε_2}. We rank the pairs (similar to *SkyRank*) and calculate the F-measure in each skyline in lines 12-19. We find the cut-off that yields the highest F-measure in line 22. Finally, we return the preference function p and the cut-off c_t.
The labeling of the pairs in the test set is then straightforward (see Alg. D.2 in paper D). We rank the pairs using the preference p. However, we can actually avoid ranking all the pairs, but we rank only c_t of them, given that we already know the cut-off. Finally, we label the ranked pairs as positive and the rest as negative.

5.3 *SkyEx-T* vs machine learning

In this section, we will compare the results of *SkyEx-T* to machine learning techniques in terms of model accuracy, explainability, and robustness. We will use two datasets, *North Denmark spatial entities* (North-DK) extracted as in [51] and used in [52, 53] and the *Fodor's and Zagat's restaurants*[4] (Restaurants).

Model accuracy. We compare SkyEx-T to Support Vector Machine (SVM) [116], Decision Trees [117], Random Forest [131], Extremely Randomized Trees (Extra Trees) [132], Extreme Gradient Boosted Trees (XGBoost) [133], Multi Layer Perceptron (MLP) [134]. All the methods use the *LGM-X* features. We trained on small percentages of data (0.05%-20%) for North-DK

[4]https://www.cs.utexas.edu/users/ml/riddle/data.html

and and (1%-20%) for Restaurants. *SkyEx-T* yields highest F-measure than the machine learning techniques for very small training sets (0.05%, 0.1%, 0.4%, and 4%) in North-DK (Fig. 1.5). In Restaurants (Fig. 1.6), *SkyEx-T* is amongst the three best approaches for 1% and 4%, while SVM, MLP, and XGBoost fail on tiny traing sets. On average, the difference of *SkyEx-T* F-measure from the maximal F-measure for training sets up to 20% of the data is only 0.83% in North-DK and 4.85% in Restaurants. *Despite its light training, SkyEx-T is able to guarantee machine-learning-level of accuracy overall, and on very small training sets, to outperform them*. Moreover, there was no method with consistently best F-measure for all training sets.

Training size	**0.05%**	**0.10%**	**0.40%**	**0.80%**	**1%**	**4%**	**8%**	**12%**	**16%**	**20%**	**80%**
					F-measure						
SVM	0.655	0.653	0.683	0.692	0.694	0.708	0.713	0.715	0.718	0.719	0.723
DecisionTree	0.596	0.589	0.609	0.613	0.612	0.622	0.632	0.634	0.641	0.644	0.667
RandomForest	0.678	0.682	0.696	0.702	0.700	0.715	0.721	0.725	0.727	0.730	**0.749**
ExtraTrees	0.670	0.676	0.693	0.700	0.699	0.710	0.717	0.721	0.723	0.726	0.744
XGBoost	0.673	0.679	0.700	0.705	0.704	0.717	**0.724**	**0.728**	0.731	**0.733**	0.747
MLP	0.678	0.688	**0.708**	**0.719**	**0.709**	0.719	0.719	0.724	**0.731**	0.724	0.727
SkyEx-T	**0.682**	**0.690**	**0.708**	0.705	0.706	**0.736**	0.717	0.718	0.711	0.711	0.727
					Difference from Max F-measure in %						
SVM	3.96%	5.36%	3.53%	3.76%	2.12%	3.80%	1.52%	1.79%	1.78%	1.91%	3.47%
DecisionTree	12.61%	14.64%	13.98%	14.74%	13.68%	15.49%	12.71%	12.91%	12.31%	12.14%	10.95%
RandomForest	0.59%	1.16%	1.69%	2.36%	1.27%	2.85%	0.41%	0.41%	0.55%	0.41%	0.00%
ExtraTrees	1.76%	2.03%	2.12%	2.64%	1.41%	3.53%	0.97%	0.96%	1.09%	0.95%	0.67%
XGBoost	1.32%	1.59%	1.13%	1.95%	0.71%	2.58%	0.00%	0.00%	0.00%	0.00%	0.27%
MLP	0.59%	0.29%	0.00%	0.00%	0.00%	2.31%	0.69%	0.55%	0.00%	1.23%	2.94%
SkyEx-T	0.00%	0.00%	0.00%	1.95%	0.42%	0.00%	0.97%	1.37%	2.74%	3.00%	2.94%

Table 1.5: SkyEx-T versus Machine Learning on North-DK (Reproduced from [55])

Model explainability. The explainability of the model is fundamental for many business applications, and "the right to explanation" is required from EU's General Data Protection Regulation (GDPR) [135]. *SkyEx-T* has a high explainability because we can see the features, which feature direction is preferred, or which features are preferred over others. A *SkyEx* example model can look like this:
(high(SimName) $\triangle$ *high(LGM_baseScore)* $\triangle$ *high(SimAddress))* $\triangleright$ *(high(Sorted_Dice_bigrams)* $\triangle$ *high(Dice_bigrams)* $\triangle$ *high(Sorted_Soft_Jaccard)* $\triangle$ *high(LGM_Dice_bigrams))*.
The machine learning techniques in Tables 1.5 and 1.6 have little or no explainability. Decision Trees can be explainable when the depth is small, and the number of features is low, but in our case, the depth went up to 6 for Restaurants and up to 42 for North-DK, making them inexplainable in practice. Moreover, Decision Trees were always outperformed by most of the other algorithms (Tables D.3 and D.4). The rest of the methods are far from explainable. *In practice, SkyEx-T has a much better model explainability when*

Training size	1%	4%	8%	12%	16%	20%	80%
	F-measure						
SVM	0.196	0.777	0.847	0.846	0.858	0.875	0.889
DecisionTree	0.818	0.798	0.796	0.810	0.831	0.816	0.875
RandomForest	0.743	0.830	0.843	0.844	0.843	0.859	0.879
ExtraTrees	**0.823**	**0.836**	**0.857**	0.853	0.860	**0.885**	0.904
XGBoost	0.000	0.724	0.823	0.827	0.847	0.870	**0.910**
MLP	0.077	0.789	0.837	**0.877**	**0.870**	0.874	0.871
SkyEx-T	0.782	0.813	0.831	0.823	0.821	0.828	0.820
	Difference from Max F-measure in %						
SVM	76.23%	7.13%	1.23%	3.46%	1.41%	1.07%	2.30%
DecisionTree	0.56%	4.59%	7.16%	7.61%	4.56%	7.79%	3.84%
RandomForest	9.74%	0.77%	1.66%	3.67%	3.14%	2.88%	3.41%
ExtraTrees	0.00%	0.00%	0.00%	2.65%	1.18%	0.00%	0.66%
XGBoost	100.00%	13.46%	4.06%	5.61%	2.72%	1.70%	0.00%
MLP	90.62%	5.62%	2.40%	0.00%	0.00%	1.26%	4.22%
SkyEx-T	4.98%	2.78%	3.12%	6.09%	5.70%	6.41%	9.92%

Table 1.6: SkyEx-T versus Machine Learning on Restaurants (Reproduced from [55])

compared to all the machine learning techniques.

Model configuration and robustness. *SkyEx-T* preference function has no coefficients or deep architecture; consequently, it does not need new tuning and reconfiguration. On the contrary, we might need to tune again the parameters and hyperparameters of the machine learning techniques every time we need to deploy them on new data. *In terms of robustness in deployment, SkyEx-T shows to be robust, while the machine learning techniques might turn out to be fragile in a real deployment.*

6 Contributions of the thesis

The overall objective of this thesis is to propose algorithms and tools that aid in obtaining large, duplicate-free, rich-in-attribute spatial entity datasets. To obtain a larger dataset, we addressed the problem of spatial data extraction first in Paper A [51], where we contributed with effective data extraction algorithms that maximize the amount of data extracted. For high quality, duplicate-free, and rich-in-attributes dataset, we studied the problem of spatial entity linkage in Paper B [53], C [54], D [55], and E [52], which aims that finding duplicate records of the same physical entity and improve the quality of the data. We can summarize the thesis contributions as follows:
Paper A [51] is the first to explicitly address the geo-social data extraction problem from social networks and online data sources, and more specifically,

maximizing the data extracted while minimizing the number of requests. We provide a summary of each source's API limitations and propose multi-source seed-driven (MSSD) algorithms that use the points of one source as a seed for the rest of the sources. Additionally, we further optimize the proposed solutions to only use about 16% of the requests while obtaining around 90% of the data. We experimentally demonstrate that a multi-source seed-driven solution is needed for the algorithm not to converge fast and continue obtaining new data.

Paper E [52] tackles the problem of spatial entity linkage and contributes with two novel algorithms: *QuadFlex*, a spatial blocking technique inspired by a quadtree but flexible to allow assigning points in more than one child not to miss relevant comparisons, and *SkyEx*, a skyline-based algorithm for classifying the pairs. To the best of our knowledge, this paper is the first to use skylines and Pareto optimality for classifying pairs in entity linkage.

Paper B [53] extends Paper E [52] with three more algorithms *SkyEx-F*, *SkyEx-FES*, and *SkyEx-D*. *SkyEx-F* and *SkyEx-FES*, similarly to *SkyEx* use a threshold of k number of skylines to separate the classes, but are better structured for setting the cut-off. *SkyEx-FES* guarantees the results of *SkyEx-F* while stopping earlier, in only 20% of the skylines, as demonstrated experimentally. The most important contribution of this paper is the *SkyEx-D* algorithm, which is completely parameter-free and fully unsupervised, but still can achieve an F-measure that is close to the F-measure of the optimal cut-off.

Paper C [54] provides `skyex`, an R-package that offers the skyline-based algorithms wrapped in user-friendly functions calls. Additionally, `skyex` covers the whole pipeline of entity linkage and offers a powerful Analysis and Visualization module.

Paper D [55] contributes with a trained skyline-based algorithm *SkyEx-T* that needs less than 1% of the data to train on in order to achieve machine learning-level accuracy. Besides model accuracy, *SkyEx-T* outperforms the machine learning techniques in terms of model explainability and robustness. Instead of traditional similarity metrics, we extend the state-of-the-art *LGM-Sim* meta similarity function to obtain domain-specific similarity features.

The proposed algorithms are novel and show significant improvement over the state-of-the-art algorithms. Moreover, these papers draw attention to the unsolved problem of lacking fresh, good-quality geo-social data, a phenomenon that is becoming more and more obvious in the published articles of the last decade.

7 Future directions

Several interesting future directions can extend and continue the research in this thesis. Regarding data extraction, we considered only location-based API requests. Alternatively, a hybrid user-based, keyword-based and location-based approach could be of interest. The algorithm can be adapted to switch between different types of API calls while traversing the geo-social graph, aiming to maximize the data extracted. Similarly to the rationale of *MSSD** where we will pre-process the seed points together with the newly-discovered source points to find the next seed points, we could try to find the next keywords or users of interest that can result in a larger amount of data extracted.

Another future direction could be formalizing the geo-social data extraction problem not only based on maximizing the amount of data extracted but also maximizing the area covered, having a higher semantic diversity of the locations, having a larger user-base, etc. All these conditions together can aid in obtaining a better and richer dataset. Finally, we could use more complex and even supervised techniques for choosing the API parameters r and p (radius and point) for the next API query, as opposed to the current heuristic-based techniques and the unsupervised DBSCAN. Additionally, we could implement a reinforcement learning method to learn the parameters while live-querying the sources.

The spatial entity linkage research can be extended with hybrid blocking techniques that use other attributes together with the geo-coordinates. These solutions would reduce the number of comparisons while trying not to miss relevant ones. Regarding the skyline-based solutions, we could try more complex preference functions that use weights for the different features, meaning that we would be able to express how much more we prefer a feature over another. Another interesting future direction would be classifying the pairs not simply as positive or negative, but having multiple classes representing other relationships such as one entity included in another, two entities being part of the same "mother" entity, etc. In this case, the skyline-based algorithm could be changed to use different preference functions and cut-offs for the different relationships that two entities can exhibit.

Future directions involving using a budget for obtaining some of the labels (through an Oracle or crowdsourcing) could improve the method's accuracy. The crowdsourcing or the Oracle will provide the labels of a sample of pairs and given these labeled data, we will try to infer and improve the overall labeling of the rest of the pairs. These solutions will need to focus on managing the budget by choosing the pairs that will assist in revealing other labels in the dataset; for example, choosing pairs that will trigger labeling other pairs through transitivity (if $\langle A, B \rangle$ and $\langle B, C \rangle$ are the same entity, so it is $\langle B, C \rangle$),

clustering the pairs and inferring the label on the whole cluster, using the skylines to infer the label on the whole skyline, etc.
As for `skyex`, we consider extending it with a Training module that can accommodate the new *SkyEx-T* algorithm. Besides the R environment, we plan to provide a similar Python package, given that Python is also a commonly-used environment in data science.

References

[1] R. Lee and K. Sumiya, "Measuring geographical regularities of crowd behaviors for twitter-based geo-social event detection," in *Proceedings of the 2nd ACM SIGSPATIAL international workshop on location based social networks*, 2010, pp. 1–10.

[2] L. Ferrari, A. Rosi, M. Mamei, and F. Zambonelli, "Extracting urban patterns from location-based social networks," in *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, 2011, pp. 9–16.

[3] S. Scellato, A. Noulas, and C. Mascolo, "Exploiting place features in link prediction on location-based social networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 1046–1054.

[4] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 1082–1090.

[5] C. Zhang, L. Shou, K. Chen, G. Chen, and Y. Bei, "Evaluating geo-social influence in location-based social networks," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012, pp. 1442–1451.

[6] M. A. Saleem, R. Kumar, T. Calders, X. Xie, and T. B. Pedersen, "Location influence in location-based social networks," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 2017, pp. 621–630.

[7] T. Zhou, J. Cao, B. Liu, S. Xu, Z. Zhu, and J. Luo, "Location-based influence maximization in social networks," in *Proceedings of the 24th ACM international on conference on information and knowledge management*, 2015, pp. 1211–1220.

[8] J. Bao, Y. Zheng, and M. F. Mokbel, "Location-based and preference-aware recommendation using sparse geo-social networking data," in *Proceedings of the 20th international conference on advances in geographic information systems*, 2012, pp. 199–208.

[9] H. Wang, M. Terrovitis, and N. Mamoulis, "Location recommendation in location-based social networks using user check-in data," in *Proceedings of the 21st ACM SIGSPATIAL international conference on advances in geographic information systems*, 2013, pp. 374–383.

[10] B. Liu and H. Xiong, "Point-of-interest recommendation in location based social networks with topic and location awareness," in *Proceedings of the 2013 SIAM international conference on data mining*. SIAM, 2013, pp. 396–404.

[11] H. Gao, J. Tang, X. Hu, and H. Liu, "Exploring temporal effects for location recommendation on location-based social networks," in *Proceedings of the 7th ACM conference on Recommender systems*, 2013, pp. 93–100.

[12] G. Ference, M. Ye, and W.-C. Lee, "Location recommendation for out-of-town users in location-based social networks," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013, pp. 721–726.

[13] J.-D. Zhang and C.-Y. Chow, "igslr: personalized geo-social location recommendation: a kernel density estimation approach," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2013, pp. 334–343.

[14] N. Li and G. Chen, "Analysis of a location-based social network," in *2009 international conference on computational science and engineering*, vol. 4. Ieee, 2009, pp. 263–270.

[15] S. Scellato, C. Mascolo, M. Musolesi, and V. Latora, "Distance matters: Geo-social metrics for online social networks." in *WOSN*, 2010.

[16] N. Armenatzoglou, S. Papadopoulos, and D. Papadias, "A general framework for geo-social query processing," *Proceedings of the VLDB Endowment*, vol. 6, no. 10, pp. 913–924, 2013.

[17] D. J. Crandall, L. Backstrom, D. Cosley, S. Suri, D. Huttenlocher, and J. Kleinberg, "Inferring social ties from geographic coincidences," *Proceedings of the National Academy of Sciences*, vol. 107, no. 52, pp. 22 436–22 441, 2010.

[18] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo, "A random walk around the city: New venue recommendation in location-based social

networks," in *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*. Ieee, 2012, pp. 144–153.

[19] G. Li, S. Chen, J. Feng, K.-l. Tan, and W.-S. Li, "Efficient location-aware influence maximization," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 87–98.

[20] T. Emrich, M. Franzke, N. Mamoulis, M. Renz, and A. Züfle, "Geo-social skyline queries," in *International Conference on Database Systems for Advanced Applications*. Springer, 2014, pp. 77–91.

[21] H. Gao, J. Tang, X. Hu, and H. Liu, "Content-aware point of interest recommendation on location-based social networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 29, no. 1, 2015.

[22] S. Feng, X. Li, Y. Zeng, G. Cong, and Y. M. Chee, "Personalized ranking metric embedding for next new poi recommendation," in *IJCAI'15 Proceedings of the 24th International Conference on Artificial Intelligence*. ACM, 2015, pp. 2069–2075.

[23] D. Jurgens, T. Finethy, J. McCorriston, Y. Xu, and D. Ruths, "Geolocation prediction in twitter using social networks: A critical analysis and review of current practice," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 9, no. 1, 2015.

[24] M. Weiler, K. A. Schmid, N. Mamoulis, and M. Renz, "Geo-social co-location mining," in *Second International ACM Workshop on Managing and Mining Enriched Geo-Spatial Data*, 2015, pp. 19–24.

[25] H. Yin, Z. Hu, X. Zhou, H. Wang, K. Zheng, Q. V. H. Nguyen, and S. Sadiq, "Discovering interpretable geo-social communities for user behavior prediction," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 2016, pp. 942–953.

[26] Z. Yao, Y. Fu, B. Liu, Y. Liu, and H. Xiong, "Poi recommendation: A temporal matching between poi popularity and user regularity," in *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE, 2016, pp. 549–558.

[27] D. Hristova, M. J. Williams, M. Musolesi, P. Panzarasa, and C. Mascolo, "Measuring urban social diversity using interconnected geo-social networks," in *Proceedings of the 25th international conference on world wide web*, 2016, pp. 21–30.

[28] S. Zhao, T. Zhao, H. Yang, M. Lyu, and I. King, "Stellar: Spatial-temporal latent ranking for successive point-of-interest recommendation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[29] J. L. Z. Cai, M. Yan, and Y. Li, "Using crowdsourced data in location-based social networks to explore influence maximization," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[30] X. Wang, Y. Zhang, W. Zhang, and X. Lin, "Efficient distance-aware influence maximization in geo-social networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 3, pp. 599–612, 2016.

[31] J. Li, T. Sellis, J. S. Culpepper, Z. He, C. Liu, and J. Wang, "Geo-social influence spanning maximization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 8, pp. 1653–1666, 2017.

[32] F. Yu, N. Che, Z. Li, K. Li, and S. Jiang, "Friend recommendation considering preference coverage in location-based social networks," in *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 2017, pp. 91–105.

[33] Q. Zhu, H. Hu, C. Xu, J. Xu, and W.-C. Lee, "Geo-social group queries with minimum acquaintance constraints," *The VLDB Journal*, vol. 26, no. 5, pp. 709–727, 2017.

[34] D. Wu, J. Shi, and N. Mamoulis, "Density-based place clustering using geo-social network data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 5, pp. 838–851, 2017.

[35] J. Zhao, Y. Gao, G. Chen, C. S. Jensen, R. Chen, and D. Cai, "Reverse top-k geo-social keyword queries in road networks," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 2017, pp. 387–398.

[36] P.-P. Zhao, H.-F. Zhu, Y. Liu, Z.-T. Zhou, Z.-X. Li, J.-J. Xu, L. Zhao, and V. S. Sheng, "A generative model approach for geo-social group recommendation," *Journal of Computer Science and Technology*, vol. 33, no. 4, pp. 727–738, 2018.

[37] R. Gao, J. Li, X. Li, C. Song, and Y. Zhou, "A personalized point-of-interest recommendation model via fusion of geo-social information," *Neurocomputing*, vol. 273, pp. 159–170, 2018.

[38] X. Wu, L. Fu, Y. Yao, X. Fu, X. Wang, and G. Chen, "Glp: A novel framework for group-level location promotion in geo-social networks,"

IEEE/ACM Transactions on Networking, vol. 26, no. 6, pp. 2870–2883, 2018.

[39] D. Yang, B. Qu, J. Yang, and P. Cudre-Mauroux, "Revisiting user mobility and social relationships in lbsns: a hypergraph embedding approach," in *The World Wide Web Conference*, 2019, pp. 2147–2157.

[40] L. Wang, Z. Yu, F. Xiong, D. Yang, S. Pan, and Z. Yan, "Influence spread in geo-social networks: a multiobjective optimization perspective," *IEEE Transactions on Cybernetics*, 2019.

[41] K. Yao, D. Papadias, and S. Bakiras, "Density-based community detection in geo-social networks," in *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*, 2019, pp. 110–119.

[42] W. Li and S. Zlatanova, "Effective geo-social group detection in location-based social networks," in *2019 IEEE International Symposium on Multimedia (ISM)*. IEEE, 2019, pp. 247–2477.

[43] M. A. Saleem, R. Kumar, T. Calders, and T. B. Pedersen, "Effective and efficient location influence mining in location-based social networks," *Knowledge and Information Systems*, vol. 61, no. 1, pp. 327–362, 2019.

[44] Z. Pan, L. Cui, X. Wu, Z. Zhang, X. Li, and G. Chen, "Deep potential geo-social relationship mining for point-of-interest recommendation," *IEEE Access*, vol. 7, pp. 99 496–99 507, 2019.

[45] X. Xiong, S. Qiao, Y. Li, N. Han, G. Yuan, and Y. Zhang, "A point-of-interest suggestion algorithm in multi-source geo-social networks," *Engineering Applications of Artificial Intelligence*, vol. 88, p. 103374, 2020.

[46] D. Yang, B. Qu, J. Yang, and P. Cudre-Mauroux, "Lbsn2vec++: Heterogeneous hypergraph embedding for location-based social networks," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[47] L. Chen, C. Liu, R. Zhou, J. Xu, J. X. Yu, and J. Li, "Finding effective geo-social group for impromptu activities with diverse demands," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 698–708.

[48] K. Wang, S. Wang, X. Cao, and L. Qin, "Efficient radius-bounded community search in geo-social networks," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[49] T. Anwar, K. Liao, A. Goyal, T. Sellis, A. Kayes, and H. Shen, "Inferring location types with geo-social-temporal pattern mining," *IEEE Access*, vol. 8, pp. 154 789–154 799, 2020.

[50] P. Jin, Z. Liu, and Y. Xiao, "Discovering the most influential geo-social object using location based social network data," in *2020 IEEE International Conference on Knowledge Graph (ICKG)*. IEEE, 2020, pp. 607–614.

[51] S. Isaj and T. B. Pedersen, "Seed-driven geo-social data extraction," in *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*, 2019, pp. 11–20.

[52] S. Isaj, E. Zimányi, and T. B. Pedersen, "Multi-source spatial entity linkage," in *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*, 2019, pp. 1–10.

[53] S. Isaj, T. B. Pedersen, and E. Zimányi, "Multi-source spatial entity linkage," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[54] S. Isaj and T. B. Pedersen, "skyex: an r package for entity linkage," in *International Conference on Extending Database Technology*, 2020, pp. 587–590.

[55] S. Isaj, V. Kaffes, T. B. Pedersen, and G. Giannopoulos, "Explainable and robust skyline-based spatial entity linkage on tiny training data," in *Under submission*, 2021.

[56] S. Isaj, N. B. Seghouani, and G. Quercini, "Profile reconciliation through dynamic activities across social networks," in *International Conference on Advanced Information Systems Engineering*. Springer, 2019, pp. 126–141.

[57] S. H. Lee, P.-J. Kim, and H. Jeong, "Statistical properties of sampled networks," *Physical review E*, vol. 73, no. 1, p. 016102, 2006.

[58] D. Preoţiuc-Pietro and T. Cohn, "Mining user behaviours: a study of check-in patterns in location based social networks," in *Proceedings of the 5th annual ACM web science conference*, 2013, pp. 306–315.

[59] N. Bennacer, F. Bugiotti, M. Hewasinghage, S. Isaj, and G. Quercini, "Interpreting reputation through frequent named entities in twitter," in *International Conference on Web Information Systems Engineering*. Springer, 2017, pp. 49–56.

[60] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 591–600.

[61] N. B. Seghouani, F. Bugiotti, M. Hewasinghage, S. Isaj, and G. Quercini, "A frequent named entities-based approach for interpreting reputation in twitter," *Data Science and Engineering*, vol. 3, no. 2, pp. 86–100, 2018.

[62] M. Lenzerini, "Data integration: A theoretical perspective," in *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2002, pp. 233–246.

[63] X. L. Dong and D. Srivastava, "Big data integration," in *2013 IEEE 29th international conference on data engineering (ICDE)*. IEEE, 2013, pp. 1245–1248.

[64] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis, and K. Stefanidis, "An overview of end-to-end entity resolution for big data," *ACM Computing Surveys (CSUR)*, vol. 53, no. 6, pp. 1–42, 2020.

[65] D. Firmani, B. Saha, and D. Srivastava, "Online entity resolution using an oracle," *Proceedings of the VLDB Endowment*, vol. 9, no. 5, pp. 384–395, 2016.

[66] K. Shu, S. Wang, J. Tang, R. Zafarani, and H. Liu, "User identity linkage across online social networks: A review," *Acm Sigkdd Explorations Newsletter*, vol. 18, no. 2, pp. 5–17, 2017.

[67] C. T. Yui, L. J. Liang, W. J. Soon, and W. Husain, "A survey on data integration in bioinformatics," in *International Conference on Informatics Engineering and Information Science*. Springer, 2011, pp. 16–28.

[68] P. Christen, T. Churches, and M. Hegland, "Febrl–a parallel open source data linkage system," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2004, pp. 638–647.

[69] R. Maskat, N. W. Paton, and S. M. Embury, "Pay-as-you-go configuration of entity resolution," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXIX*. Springer, 2016, pp. 40–65.

[70] J. Efremova, B. Ranjbar-Sahraei, H. Rahmani, F. A. Oliehoek, T. Calders, K. Tuyls, and G. Weiss, "Multi-source entity resolution for genealogical data," in *Population reconstruction*. Springer, 2015, pp. 129–154.

[71] X. L. Dong and D. Srivastava, "Big data integration," *Synthesis Lectures on Data Management*, vol. 7, no. 1, pp. 1–198, 2015.

[72] G. Papadakis, E. Ioannou, C. Niederée, T. Palpanas, and W. Nejdl, "Eliminating the redundancy in blocking-based entity resolution methods," in *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, 2011, pp. 85–94.

[73] ——, "Beyond 100 million entities: large-scale blocking-based resolution for heterogeneous data," in *Proceedings of the fifth ACM international conference on Web search and data mining*, 2012, pp. 53–62.

[74] G. Papadakis, J. Svirsky, A. Gal, and T. Palpanas, "Comparative analysis of approximate blocking techniques for entity resolution," *Proceedings of the VLDB Endowment*, vol. 9, no. 9, pp. 684–695, 2016.

[75] G. Papadakis, E. Ioannou, T. Palpanas, C. Niederee, and W. Nejdl, "A blocking framework for entity resolution in highly heterogeneous information spaces," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 12, pp. 2665–2682, 2012.

[76] V. Efthymiou, K. Stefanidis, and V. Christophides, "Big data entity resolution," in *2015 IEEE International Conference on Big Data (IEEE BigData 2015)*, 2015.

[77] ——, "Benchmarking blocking algorithms for web entities," *IEEE Transactions on Big Data*, 2016.

[78] A. Panchenko, D. Babaev, and S. Obiedkov, "Large-scale parallel matching of social network profiles," in *International Conference on Analysis of Images, Social Networks and Texts*. Springer, 2015, pp. 275–285.

[79] G. Quercini, N. Bennacer, M. Ghufran, and C. N. Jipmo, "Liaison: reconciliation of individuals profiles across social networks," in *Advances in Knowledge Discovery and Management*. Springer, 2017, pp. 229–253.

[80] R. Karam, F. Favetta, R. Kilany, and R. Laurini, "Integration of similar location based services proposed by several providers," in *International Conference on Networked Digital Technologies*. Springer, 2010, pp. 136–144.

[81] M. Edwards, S. Wattam, P. Rayson, and A. Rashid, "Sampling labelled profile data for identity resolution," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 540–547.

[82] O. Peled, M. Fire, L. Rokach, and Y. Elovici, "Entity matching in online social networks," in *2013 International Conference on Social Computing*. IEEE, 2013, pp. 339–344.

[83] O. Goga, H. Lei, S. H. K. Parthasarathi, G. Friedland, R. Sommer, and R. Teixeira, "Exploiting innocuous activity for correlating users across sites," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 447–458.

[84] M. Magnani and D. Montesi, "A survey on uncertainty management in data integration," *Journal of Data and Information Quality (JDIQ)*, vol. 2, no. 1, pp. 1–33, 2010.

[85] M. Schäfers and U. W. Lipeck, "Simmatching: adaptable road network matching for efficient and scalable spatial data integration," in *Proceedings of the 1st ACM SIGSPATIAL PhD Workshop*, 2014, pp. 1–5.

[86] R. Abdalla, "Geospatial data integration," in *Introduction to Geospatial Information and Communication Technology (GeoICT)*. Springer, 2016, pp. 105–124.

[87] P. Tabarro, J. Pouliot, R. Fortier, and L.-M. Losier, "A webgis to support gpr 3d data acquisition: A first step for the integration of underground utility networks in 3d city models," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, p. 43, 2017.

[88] S. Balley, C. Parent, and S. Spaccapietra, "Modelling geographic data with multiple representations," *International Journal of Geographical Information Science*, vol. 18, no. 4, pp. 327–352, 2004.

[89] V. Walter and D. Fritsch, "Matching spatial data sets: a statistical approach," *International Journal of geographical information science*, vol. 13, no. 5, pp. 445–473, 1999.

[90] L. Shu, A. Chen, M. Xiong, and W. Meng, "Efficient spectral neighborhood blocking for entity resolution," in *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 2011, pp. 1067–1078.

[91] A. Morana, T. Morel, B. Berjawi, and F. Duchateau, "Geobench: a geospatial integration tool for building a spatial entity matching benchmark," in *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2014, pp. 533–536.

[92] E. Moreau, F. Yvon, and O. Cappé, "Robust similarity measures for named entities matching," in *COLING 2008, 22nd International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, UK*, 2008, pp. 593–600.

[93] R. Santos, P. Murrieta-Flores, and B. Martins, "Learning to combine multiple string similarity metrics for effective toponym matching," *International Journal of Digital Earth*, vol. 11, 2018. [Online]. Available: https://doi.org/10.1080/17538947.2017.1371253

[94] C. Davis Jr and E. Salles, "Approximate string matching for geographic names and personal names," 01 2007, pp. 49–60.

[95] D. Kılınç, "An accurate toponym-matching measure based on approximate string matching," *Journal of Information Science*, vol. 42, pp. 138–149, 2016. [Online]. Available: https://doi.org/10.1177/0165551515590097

[96] V. Kaffes, G. Giannopoulos, N. Karagiannakis, and N. Tsakonas, "Learning domain specific models for toponym interlinking," in *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2019, pp. 504–507.

[97] G. Recchia and M. Louwerse, "A comparison of string similarity measures for toponym matching," *COMP 2013 - ACM SIGSPATIAL International Workshop on Computational Models of Place*, pp. 54–61, 11 2013.

[98] G. Giannopoulos, V. Kaffes, and G. Kostoulas, "Learning advanced similarities and training features for toponym interlinking," in *European Conference on Information Retrieval*. Springer, 2020, pp. 111–125.

[99] N. Dalvi, M. Olteanu, M. Raghavan, and P. Bohannon, "Deduplicating a places database," in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 409–418. [Online]. Available: https://doi.org/10.1145/2566486.2568034

[100] R. Santos, P. Murrieta-Flores, P. Calado, and B. Martins, "Toponym matching through deep neural networks," *International Journal of Geographical Information Science*, vol. 32, pp. 324–348, 2018. [Online]. Available: https://doi.org/10.1080/13658816.2017.1390119

[101] K. Alexis, V. Kaffes, and G. Giannopoulos, "Boosting toponym interlinking by paying attention to both machine and deep learning," ser. GeoRich '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3403896.3403970

[102] V. Sehgal, L. Getoor, and P. D. Viechnicki, "Entity resolution in geospatial data integration," in *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, 2006, pp. 83–90.

[103] G. Giannopoulos and M. Meimaris, "Learning domain driven and semantically enriched embeddings for poi classification," in *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*, 2019, pp. 214–217.

[104] B. Berjawi, E. Chesneau, F. Duchateau, F. Favetta, C. Cunty, M. Miquel, and R. Laurini, "Representing uncertainty in visual integration." in *DMS*, 2014, pp. 365–371.

[105] A.-M. O. Raimond and S. Mustière, "Data matching–a matter of belief," in *Headway in spatial data handling*. Springer, 2008, pp. 501–519.

[106] S. Isaj and T. B. Pedersen, "Seed-driven geo-social data extraction–full version," *arXiv preprint arXiv:1901.06712*, 2019.

[107] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[108] H. Samet, "The quadtree and related hierarchical data structures," *ACM Computing Surveys (CSUR)*, vol. 16, no. 2, pp. 187–260, 1984.

[109] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *ACL*, 1994.

[110] C. Fellbaum, "Wordnet," in *Theory and applications of ontology: computer applications*. Springer, 2010, pp. 231–243.

[111] Y. Censor, "Pareto optimality in multiobjective problems," *Applied Mathematics and Optimization*, vol. 4, no. 1, pp. 41–59, 1977.

[112] J. L. Bentley, D. F. Stanat, and E. H. Williams Jr, "The complexity of finding fixed-radius near neighbors," *Information processing letters*, vol. 6, no. 6, pp. 209–212, 1977.

[113] H. Köpcke, A. Thor, and E. Rahm, "Evaluation of entity resolution approaches on real-world match problems," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 484–493, 2010.

[114] G.-w. You, S.-w. Hwang, Z. Nie, and J.-R. Wen, "Socialsearch: enhancing entity search with social network matching," in *Proceedings of the 14th International Conference on Extending Database Technology*, 2011, pp. 515–519.

[115] D. W. Hosmer and S. Lemeshow, "S.(1989) applied logistic regression," *John Wiley&Sons*.

[116] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[117] W. A. Belson, "Matching and prediction on the principle of biological classification," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 8, no. 2, pp. 65–75, 1959.

[118] T. Bayes, "Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s," *Philos Trans R Soc Lond*, 1763.

[119] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[120] L. Kaufman and P. Rousseeuw, "Clustering by means of medoids," 1987.

[121] T. Hastie, R. Tibshirani, and J. Friedman, "Hierarchical clustering," *EoSL*, 2009.

[122] A. Elmagarmid, I. F. Ilyas, M. Ouzzani, J.-A. Quiané-Ruiz, N. Tang, and S. Yin, "Nadeef/er: Generic and interactive entity resolution," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 1071–1074.

[123] L. Kolb, A. Thor, and E. Rahm, "Dedoop: Efficient deduplication with hadoop," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 1878–1881, 2012.

[124] X. Ke, M. Teo, A. Khan, and V. K. Yalavarthi, "A demonstration of perc: probabilistic entity resolution with crowd errors," *Proceedings of the VLDB Endowment*, vol. 11, no. 12, pp. 1922–1925, 2018.

[125] A. Ebaid, S. Thirumuruganathan, W. G. Aref, A. Elmagarmid, and M. Ouzzani, "Explainer: entity resolution explanations," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 2000–2003.

[126] E. C. Dragut, M. Ouzzani, A. K. Elmagarmid, W. G. Aref *et al.*, "Orlf: A flexible framework for online record linkage and fusion," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 2016, pp. 1378–1381.

[127] P. Konda, S. Das, P. Suganthan GC, A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. Naughton *et al.*, "Magellan: Toward building entity matching management systems," *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 1197–1208, 2016.

[128] Y. Govind, E. Paulson, P. Nagarajan, G. Paul Suganthan, A. Doan, Y. Park, G. Fung, D. Conathan, M. Carter, and M. Sun, "Cloudmatcher: a hands-off cloud/crowd service for entity matching," 2018.

[129] T. M. Cover and J. A. Thomas, "Information theory and statistics," *Elements of Information Theory*, vol. 1, no. 1, pp. 279–335, 1991.

[130] W. H. Chiu, "Skewness preference, risk taking and expected utility maximisation," *The Geneva Risk and Insurance Review*, vol. 35, no. 2, pp. 108–129, 2010.

[131] L. Breiman, "Random forests," vol. 45, no. 1, 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[132] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[133] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[134] R. Collobert and S. Bengio, "Links between perceptrons, mlps and svms," in *Proceedings of the Twenty-First International Conference on Machine Learning*, ser. ICML '04. Association for Computing Machinery, 2004, p. 23. [Online]. Available: https://doi.org/10.1145/1015330.1015415

[135] M. E. Kaminski, "The right to explanation, explained," *Berkeley Tech. LJ*, vol. 34, p. 189, 2019.

Part II

Papers

Paper A

Seed-Driven Geo-Social Data Extraction

Suela Isaj, Torben Bach Pedersen

The paper has been published in the
16th International Symposium on Spatial and Temporal Databases (SSTD'19)
August 19–21, 2019, Vienna, Austria, DOI: 10.1145/3340964.3340973

Abstract

Geo-social data has been an attractive source for a variety of problems such as mining mobility patterns, link prediction, location recommendation, and influence maximization. However, new geo-social data is increasingly unavailable and suffers several limitations. In this paper, we aim to remedy the problem of effective data extraction from geo-social data sources. We first identify the limitations of extracting geo-social data. To overcome the limitations, we propose a novel seed-driven approach that uses the points of one source as the seed to feed as queries for the others. We additionally handle differences between, and dynamics within the sources by proposing three variants for optimizing search radius. Furthermore, we provide an optimization based on recursive clustering to minimize the number of requests and an adaptive procedure to learn the specific data distribution of each source. Our comprehensive experiments with six popular sources show that our seed-driven approach yields 14.3 times more data overall, while our request-optimized algorithm retrieves up to 95% of the data with less than 16% of the requests. Thus, our proposed seed-driven approach set new standards for effective and efficient extraction of geo-social data.

1 Introduction

Each year social networks experience a continuous growth of 13% in the number of users (http://wearesocial.com/uk /blog/2018/01/ global-digital-report-2018). Consequently, more and more information is available regarding the activity that the users share, events in which they participate and the new connections they make. When data collected by social networks contain social connections (friendship links, mentions, and tags in posts, etc) as well as geographic information (check-ins, geo-data in posts and implicit location detection), then this data is usually referred as *geo-social data*. *Geo-social data* have attracted studies regarding location prediction, location recommendation, location-based advertisement, urban behavior, etc. The primary sources of geo-social data are *location-based social networks* (LBSNs) such as Gowalla, Brightkite, and Foursquare, which contain social ties, check-ins, tips and detailed information about locations. However, Gowalla and Brightkite were closed in 2012, whereas Foursquare has blocked the extraction of check-ins from its API (Application Programming Interface - set of functions and procedures that allow data extraction from a source). Other secondary sources of geo-social data are *social networks* such as Facebook, Twitter, Flickr, etc. Social networks are characterized by richness and variety of data, making them an attractive source for data extraction. However, the percentage of geo-located posts reported in the literature is less than 1% ([1–3]). Furthermore, they provide rich information about users, their networks, their activities but only a few details about locations (only the coordinates). Another less common source of location data (not necessarily geo-social) are *directories* such as Yelp, Google Places, TripAdvisor, etc, which contain locations with details such as name, phone, type of business, etc and sometimes accompanied by user reviews. In the majority of the cases, directories do not contain user profiles; even when they do, the API does not provide functions to extract user's information. Hence, it is necessary to use several sources in order to gain a complete dataset of geo-social data.

Not only is geo-social data scattered over several sources but the APIs of the sources are also highly restrictive regarding the number of requests, the amount and the type of data that can be extracted, etc. Instead of extracting the data, publicly available datasets can be used. However, their usability is limited because sometimes they lack the details about users' profiles or the locations, which could be of interest for the research purpose. Besides, the check-ins/photos/posts/reviews are sparse and scattered all over the globe, affecting the quality of the experiments while mining frequent patterns, mobility patterns, urban behavior, etc. Enriching these datasets with the missing details is not possible because the data is anonymized, so the link with the source is lost. Even when the data is not anonymized, the datasets are old

(2008-2013) and they can not map to the existing users or locations of nowadays. When we analyzed 32 papers from 2009 to 2018 using geo-social data, we found that no less than 50% used datasets that are 3-8 years older than the published article (see Appendix A in [4]).
To sum up, geo-social data is becoming even more needed and even less accessible. We thus, address the problem of *location-based geo-social data* extraction from social networks and location-based sources. We introduce the limitations of six sources of geo-social data, namely: Flickr, Twitter, Foursquare, Google Places, Yelp, and Krak. Then, we propose a seed-driven algorithm that uses the points of the richest source (the *seed*) to extract data from the others. Later, we introduce techniques to optimize the selection of radius and seed points. Our main contributions are: (i) We provide an analysis of the current limitations of data extraction from six popular geo-social data sources. (ii) We identify and formulate the problem of *maximizing the extracted geo-social data* while *minimizing the requests* to the sources. To the best of our knowledge, we are the first to optimize the data extraction process in social networks and location-based sources. (iii) We propose a novel algorithm for data extraction that uses the points of one source as *seed* to the API requests of the others. Our *seed-driven* algorithm retrieves up to 98 times more data than the default API querying. (iv) We introduce an *optimized* version of our algorithm that minimizes the requests and ensures maximized data extraction by recursively adapting the radius and the centroid of the query region. We retrieve around 90% of the data using less than 16% of the requests.
The remainder of the paper is structured as follows: first, we describe the related work in Section 2; then, we introduce the definitions and the data extraction problem in Section 3; later, we categorize the limitations of the data extraction process and we provide preliminary results from six sources in Section 4; we continue with formalizing our proposed algorithm in Section 5; next, we test the proposed solutions through real-time querying of the sources and we compare the results in Section 6; and finally, we conclude and provide further insights on our work in Section 7.

2 Related work

Despite the growing interest in geo-social related topics, the existing related work does not focus specifically on optimizing the data extraction process. Most of the existing research uses either publicly available datasets [5–11, 11–18], crawl using the default settings of the API [19–27] or do both [28, 29]. The (sparsely described) crawling methods used in these papers can be categorized as either *user-based* crawling, *location-based* crawling or *keyword-based* querying.
User-based crawling. User-based crawling is based on querying *users* for

their data and their networks as well. A user-based crawling technique mentioned in several studies is the *Snowball* technique [30, 31]. *Snowball* requires a prior *seed* of users to start with and then, traverses the network while extracting data from the network of friends. Nonetheless, *Snowball* is biased to the high degree nodes [32] and requires a well-selected seed. Another interesting method is to track the users that post with *linked accounts* [33–35], for instance, users posting from Twitter using the check-in feature of Foursquare. Nevertheless, this method is limited only to linked accounts, whose percentage is less than 1%.

Location-based crawling. Location-based crawling requires no prior knowledge, and the extraction process can start at any time. It is based on extracting data near or within a specific area. Lee et al. [36] use a periodical querying based on points extracted from Twitter. First, Twitter is queried for initial points. Then, in a later step, other requests are performed using the initial points as query points, focusing on areas detected by the user. Thus, in each step n, the points discovered in step $n-1$ are used to perform the new queries. We will refer to this method as *Self-seed*.

Keyword-based querying. As the name suggests, the source is queried [not necessarily crawled] with a keyword to find relevant data. The keyword-based querying is widely used by the research on topic mining, opinion mining, the reputation of entities, quality of samples and several related topics [37, 38] but not for geo-social topics since querying with a keyword does not guarantee that the retrieved data will be located in the queried location. For example, querying Twitter with the keyword "Brussels" can return tweets in Brussels, tweets talking about Brussels but not located there, and even tweets about brussels sprouts.

Discussion. Obviously, the keyword-based querying is not of interest due to the noise it brings. The user-based crawling requires prior information about a seed of users and applies only to social networks. Subsequently, it leaves aside other location-based sources such as Yelp, TripAdvisor, and Google Places. Moreover, if the study is based on a region of interest, the user-based crawling results in a lot of irrelevant data because even if the seed of users is well-selected from the region of interest, there is no guarantee that the friends will check-in in the area of interest. Consequently, user-based crawling produces wasted requests. The method described by [36] has some similarities with ours because it is location-based crawling and focuses on performing requests on areas discovered previously. In comparison, our approach differs significantly because (i) instead of selecting points from a single source and querying itself, we use a *seed* source to query *multiple sources*, (ii) we minimize the number of requests performed while maximizing the data extracted (iii) our seed-driven data extraction approach does not need periodical querying; it can be run continuously and simultaneously for all the sources, resulting in faster data extraction process, compared to several months like in [36]. *To*

sum up, our data extraction approach is faster, richer, request-economic, and includes multiple sources.

3 Problem definition

The notion that we will use widely in the paper is a *location*. A location in a directory is a venue with a geographical point and additional attributes like name, category, etc. Social networks contain *activities* such as check-ins, tips, photos and tweets which are geo-tagged. We denote the locations associated with the activities as *derived locations* and for brevity, just as *locations*.

Definition A.1. *A location l is a spatial entity identified within the source by a unique identifier $id(l)$. A location l has a set of attributes $A = \{a_1, a_2...a_n\}$ accompanied by their values $\{a_1(l), a_2(l)...a_n(l)\}$. A required attribute for a location l is its geographical coordinates denoted as $p(l)$*

For example, l_1 is a tweet with id = 1234567, where A={text, user, point} and the values are {"Nice day in park", 58302, <57.04, 9.91>}. Geo-social data sources usually offer an *Application Programming Interface (API)*, which is a set of functions and procedures that allow accessing a source to obtain its data. Location-based API calls allow querying with (i) a point p and a radius r, (ii) a box $< p_1, p_2, p_3, p_4 >$ and (iii) keywords. We will not consider keyword-based querying due to the noise it brings (see Section 2). The circular querying and the rectangular querying are quite similar as long as the parameters are the same. In this work, we use querying with a point p and a radius r and refer to the searched area as $Circle(p, r)$. We define a geo-social data source formally as:

Definition A.2. *A geo-social data source S, short as* source*, consists of the (complete) set of locations $L(S)$ and a source-specific extraction function $API: \mathcal{P}x\mathcal{R}^+ \Rightarrow 2^{L(S)}$, where $\mathcal{P}$ is the domain of geographical points and $\mathcal{R}^+$ is the domain of non-negative numbers. $API(p, r)$ queries with a centroid $p \in \mathcal{P}$ and a radius $r \in \mathcal{R}$ and returns a sample of locations L_p^r, such that for each $l \in L_p^r$, $p(l) \in Circle(p, r)$ and $|L_p^r| \leq M_S$, where M_S is the maximal result size for S.*

If S is Twitter, then $L(S)$ is the complete set of tweets (all the tweets posted ever on Twitter). A request with a point and a radius $< p, r >$ will retrieve a sample L_p^r of size at most M_S of the underlying activities $L_p^r(S)$ in $Circle(p, r)$. So, if $M_S = 100$, then $L_p^r \leq 100$ locations. Given that the requests are limited, they need to be used wisely in order to retrieve the largest combined result size. For example, if the first request retrieves the locations $\{l_1, l_4, l_5, l_6\}$ and the second request retrieves $\{l_2, l_4, l_5, l_6\}$, then the second request contributed with only one new location (l_2).

Problem definition. Optimizing geo-social data extraction is the problem that given a source S_i and a number of requests n finds the sequence of pairs of point and radius $\{< p_1, r_1 >, < p_2, r_2 > ... < p_n, r_n >\}$ such that the size of $L_i = \bigcup_{j=1}^{n} L_{p_j}^{r_j}$ is maximized .

API limitations	Krak	Yelp	Google Places	Foursquare	Twitter	Flickr
Bandwidth	10K/month	5K/day	1/day (from 6/2018)	550/hour	180/15 min	3.6K/hour
Max Res. Size	100	50	20	50	100	500
Hist. Access	N/A	N/A	N/A	Full	2 weeks	Full
Supp Results	4.3%	17.3%	0.5%	0.0%	0.0%	0.0%
Complete access	yes	yes	yes	yes	1%	yes
Cost	not stated	negotiable	from 200$/month	from 599$/month	149$ - 2499$/month	not stated

Table A.1: Summary of limitations of social networks

The problem aims to obtain a good compromise between the number of requests and the number of locations L_i. The optimal solution is a combination of $\{< p_1^*, r_1^* >, < p_2^*, r_2^* > ... < p_n^*, r_n^* >\}$ such that L_i is maximal (the optimal L_i^*). Given the API limitations, trying exhaustively all possible values and combinations of p and r to find L_i^* is not feasible. Hence, we propose solutions that are based on heuristics and assumptions. Before proposing our solutions, let us first introduce the API limitations for each source.

4 Limitations of existing geo-social data sources

With regard to quantifying the limitations, we present preliminary results from querying six sources: Twitter, Flickr, Foursquare, Yelp, Google Places, and Krak. Krak is a Danish website that offers information about companies, telephone numbers, etc. In addition, Krak is part of Eniro Danmark A / S. which takes care of publishing The Yellow Pages. We queried all the sources simultaneously for the region of North Denmark during November-December 2017. With respect to gaining more data, we performed additional requests using different keywords ("restaurant", "library", "cozy", etc) as well as coordinates of the cities and towns in the region.
The API **bandwidth** refers to the number of requests allowed within a time frame. For example, Twitter allows 180 requests in 15 min; meanwhile, Krak has a time window of a month. Google Places allowed 1000 requests in a day before June 2018, and now, just one request per day. If more requests are needed, the cost is 0.7 US cents/request (first 200 USD free). In our data extraction and experiments, we fix the bandwidth of Google Places to 1000 requests in a day (the former default). The **maximal result size** is the maximal number of results returned by a single request. For instance, an API call in Flickr retrieves 250 photos, but in Google Places retrieves only 20 places. The **historical access** is related to how accessible the earlier activity is. Direc-

tories such as Yelp, Krak, and Google Places do not provide historical data; they only keep track of the current state of their locations. Foursquare provides only the current state of its venues and historical access to photos and tips by querying with venues. Flickr can go back in old photos, whereas Twitter limits the results only to the last couple of weeks. The **supplemental results** are data which do not belong in the query region, but they are still added to the API result. For example, if we search for "Zara" shop in city X, the API might return the "Zara" shop which is the closest to X but in city Y. We noticed that supplemental results are present only in directories, which aim to advertise and provide results anytime. Having **access to the complete dataset** means that the API can query the whole dataset, not just a sample. For example, the Twitter API accesses only a sample of 1% while others enable access to its complete dataset. Most social networks and directories offer free APIs at no **costs**. They also offer premium or enterprise services with monthly payment or pay-as-you-go services. While some sources have predefined pricing plans (Twitter, Foursquare, and Google Places), others offer the possibility to discuss the needs and the price (Yelp). Even though a premium service has fewer restrictions, it is still needed to keep costs down.
A summary of the limitations of social networks is presented in Table. A.1. Krak is restrictive with the bandwidth. Google Places has a very small result size and only one request per day. Flickr is promising in terms of the API limitations, while Twitter shows severe problems regarding the limitation to access historical data. Foursquare and Yelp could be considered similar in terms or limitations. The number of locations and the number of points for each source are presented in Table A.2. Krak has a leading position with almost two orders of magnitude more results than any other source, followed by Flickr, Foursquare, Yelp, Google Places, and finally, Twitter. As for the temporal density, we recorded that in Flickr there are around 17 photos per day, in Twitter 10 photos per day, in Foursquare 0.03 tips and 0.36 photos per day (See Subsection 4.2 in [4] for more details on data scarcity). To sum up, a single source queried with the default API cannot provide a rich enough dataset. In the next section, we propose a novel algorithm that uses one of the sources as *seed* to extract data from the others and is capable of obtaining up to 14.3 times more data than single source initial querying (Section 6.1).

Category	Krak	Yelp	GP	FSQ	TW	Flickr
Locations	143,073	473	380	1,097	115	4,084
Points	32,461	467	356	1,093	25	2,272

Table A.2: Locations and points per source

5 Multi-source Seed-Driven approach

Section 4 studied the limitations of data extraction and quantified the performance of each of the sources. In this section, we propose a main algorithm and several adaptions to it that lead to an effective data extraction process.

5.1 Multi-Source Seed-Driven Algorithm

We will refer to the initial default API queries as **Source Initial** (*SI*) and to the set of locations they retrieve as L_I. Having no prior knowledge of the underlying data $L(S_i)$ makes it challenging to choose which API calls to perform. However, *all the sources operating in the same region contain data that maps to the same physical world*. For example, if there is a bar in the point (56.89 9.21) in Krak, probably around this point there might be this and other locations in Yelp, Google Places, and Foursquare and even some activity such as tweets, photos or check-ins in Twitter, Flickr, and Foursquare. This means that if the *SI* of a source is rich in terms of locations, then its knowledge can be used to improve the data extraction of the other sources. Hence, we propose a *seed-driven* approach to extract locations from multiple sources. The main idea is simple; *selecting one (more complete) source as the seed and feeding the points to the rest for data extraction*.

Multi-Source Seed-Driven (*MSSD*) is a function that takes as input a set of sources $S_1, S_2 ... S_k$ and outputs their corresponding sets of locations $\{L_{S_1}, L_{S_2}, ... L_{S_k}\}$ obtained from the seed-driven approach in Alg. A.1. For example, let us suppose that the seed provides a location with coordinates (57.05, 9.92) as in Fig. A.1. We can search for locations across sources within the circle with center (57.05, 9.92) and a predefined radius. The different colors in the figure represent the different sources. We can discover three locations from the red source, two from the blue source and two from the green source. The algorithm for the seed-driven approach is presented in Alg. A.1. Selecting a good seed is important; thus, we start by getting the most complete source (with the most points) in line 4. The points in the seed indicate regions of interest and are used for the API request in the sources. So, for each point in seed (for each p in P), we query the rest of the sources except the seed source. Line 7 shows the general API call for each of the sources, which is performed in correspondence to the requirements of the source. The request takes the coordinates of p and the radius r. The search returns a set of locations L_p^r, which is unioned to our source-specific output L_S.

5.2 Optimizing the Radius

We can improve further *MSSD* by adapting the API request to the source. Even though a big radius might seem like a better solution, note that the

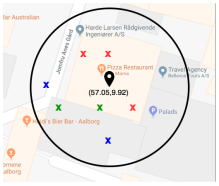
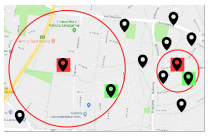


Fig. A.1: *MSSD* approach

(a) *MSSD-D* radius



(b) *MSSD-N* radius

(c) *MSSD-R* radius

Fig. A.2: *MSSD* radius

Algorithm A.1 Multi-Source Seed-Driven (*MSSD*)

Input: A set of sources $\{S_1, S_2, ...S_k\}$, radius r
Output: $\{L_{S_1}, L_{S_2}, ...L_{S_k}\}$

1: **for each** S in $\{S_1, S_2, ...S_k\}$ **do**
2: $L_S \leftarrow L_I$ $\rightarrow$/* Initialize L_S of each source with L_I*/
3: **end for**
4: Let S_{seed} be the source with the most points in $\{S_1, S_2, ...S_k\}$, L_{seed} its locations and P the distinct points in L_{seed}
5: **for each** p in P **do**
6: **for each** S in $\{S_1, S_2, ...S_k\}$ - S_{seed} **do**
7: $L_p^r \leftarrow API(p, r)$ $\rightarrow$/* API request for the source S */
8: $L_S \leftarrow L_S \cup L_p^r$
9: **end for**
10: **end for**
return $\{L_{S_1}, L_{S_2}, ...L_{S_k}\}$

API retrieves only a fixed size sample of the underlying data. Hence, if we query with points that are nearby, we might retrieve intersecting samples. We denote the maximal result size of the API for source S as M_S. Let us consider the example in Fig.A.3, where M_S is 3, which means that the API can not retrieve more than 3 tweets. If we query with a big radius as in the left part of Fig.A.3, we might get 2 out of 3 tweets in the intersection. If the radius is small, then we explore better the dense areas, but we might miss in sparser ones like in the right part of Fig. A.3. The union of tweets in both searches is just 4, where ideally it should have been 6. We propose two improvements: *using the knowledge of the seed to define the radius* and *recursively learn a suitable radius for the source*.

Multi-Source Seed-Driven Density-Based *MSSD-D*. The radius in this version is defined by the density of points in the *seed*. Before the API requests, we check the density of points in the search area in the *seed*, and we adapt the radius accordingly. Fig. A.2a illustrates the intuition behind *MSSD-D*. We are using the point in red as seed point p. Before performing any API call, we

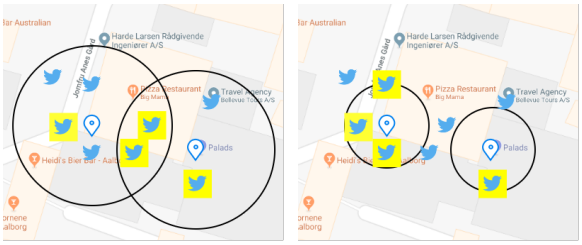


Fig. A.3: Radius adjustment

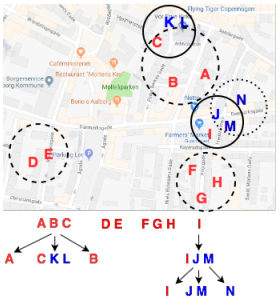


Fig. A.4: *MSSD**

check how many points of the *seed* are in the search area ($N = 4$ points in the black circle with center p and initial radius r). Second, we adjust the radius according to the density, so in this case, we divide the radius by 4 ($r_d = \frac{r}{|N|}$). Finally, we perform the API call to the source with the red circle. Alg. A.1a shows the alterations we make in Alg. A.1 for the radius calculation. We add line 5.a and 5.b after line 5 in Alg. A.1. First, we find the density of the region, and then, we adjust the radius depending on the density. We query with the adjusted $r_d = \frac{r}{|N|}$ in line 7 of Alg. A.1.

Algorithm A.1a *MSSD* Density-Based (*MSSD-D*)

5.a: Find $N = \{q | q \in Circle(p, r)\}$ →/* Find how dense the region is*/

5.b: $r_d = \frac{r}{|N|}$ →/* Adjust the radius*/

Multi-Source Seed-Driven Nearest Neighbor *MSSD-N*. As the name suggests, we use the nearest neighbor in the *seed* to define the radius. A simple illustration of this idea is presented in Fig. A.2b. For each of the points p in the seed (in red), we find the nearest neighbor q in the seed (in green), and then we query with the adjusted radius $r_n = |p - q|$. Note that we query with a small radius in dense areas and a big radius in sparse ones. Alg. A.1b instead adds line 5.a and 5.b after line 5 of Alg. A.1. It finds the nearest neighbor q of the point p. Then, we set $r_n = |p - q|$. The adjusted r_n is used to query the sources in line 7 of Alg. A.1.

Algorithm A.1b *MSSD* Nearest Neighbor (*MSSD-N*)

5.a: Find $q = \min_{q \in L_i} |p - q|$ →/* Find the nearest neighbor*/

5.b: $r_n = |p - q|$ →/* Adjust the radius*/

Multi-Source Seed-Driven Recursive *MSSD-R*. The advantage of *MSSD-D* and *MSSD-N* is that no API call is needed to adjust the radius because these calculations are performed on the *seed* points. However, there is a need for a better approach to assigning a suitable radius for a specific area. We propose a solution that adjusts the radius while querying the source. If an area not dense, we can identify it from the API call. However, in contrast to SQL queries on a database where the operations are transparent, the operations of API queries on an online source are a black box, and thus, we need to assume a certain level of transparency. Therefore, we assume that if the area contains less than M_S locations, the API call will retrieve all of them.

Assumption A.1. *For each source S in $\{S_1, S_2, ..., S_k\}$, if $Circle(p,r)$ contains $L_p^r(S)$ locations such that $|L_p^r(S)| \leq M_S$, then $API(p,r)$ will retrieve $L_p^r = L_p^r(S)$.*

The API retrieves a sample of size M_S of the underlying data in a queried region $Circle(p,r)$. If the underlying locations $L_p^r(S)$ are already less than M_S, then we assume that the API will retrieve *all* the locations lying in $Circle(p,r)$. For example, if there are 30 locations in $Circle((56.78\ 9.67), 1km)$ and $M_S = 50$, then querying with $p = (56.78\ 9.67)$ and r=1 km will return all 30 locations.

Theorem A.1. *Let $< p, r >$ be a pair of point and radius such that $API(p,r) = L_p^r$ where $|L_p^r| < M_S$. Then, for all r' such that $r' < r$, $L_p^{r'} \subseteq L_p^r$.*

Proof. Let us assume that there are $|L_p^r(S)|$ locations in $Circle(p,r)$ and $|L_p^{r\prime}(S)|$ locations in $Circle(p,r\prime)$. Since $r' < r$, then the surface covered by $Circle(p,r\prime)$ is smaller than the surface covered by $Circle(p,r)$ ($\pi {r'}^2 < \pi r^2$). Consequently, $L_p^{r'}(S) \subseteq L_p^r(S)$. According to Assumption A.1, since $API(p,r)$ retrieves $|L_p^r| < M_S$, then $L_p^r = L_p^r(S)$ and $|L_p^r(S)| < M_S$. Given that $L_p^{r'}(S) \subseteq L_p^r(S)$ and $|L_p^r(S)| < M_S$, we conclude that $|L_p^{r'}| < M_S$ and $L_p^{r'} = L_p^{r'}(S)$. Finally, from $L_p^{r'}(S) \subseteq L_p^r(S)$, we derive that $L_p^{r'} \subseteq L_p^r$. □

This is an important finding that will be used in defining *MSSD-R*. Since there are less than M_S locations retrieved by the API call in source S, there are no new locations to be gained by querying with a smaller radius. Thus, we propose a recursive method that uses Theorem A.1 as our stopping condition. First, we query with an initial large radius, and if the result size is M_S, then we know this is a dense area, and we perform another request with a smaller radius. The search stops when the number of returned results is smaller than the maximal result size because according to Theorem A.1. The recursive method is illustrated in Fig. A.2c. Let us suppose that we are querying source S and the maximal result size is $M_S = 5$. After querying with the green circle, we get 5 locations so we know that the area is dense. We reduce the radius

by α ($r_r = \frac{r}{\alpha}$) and query again with the blue circle. We get 5 locations again, therefore we continue once more with a smaller radius. When we query with the red circle, we get only 2 locations, so we stop. Alg. A.1c (*MSSD-R*) has a modification where a new algorithm is called. Instead of querying with a static r, we perform a recursive procedure to adjust the radius. Alg. A.2 takes the following parameters: the radius r_r, the coefficient α which is used to reduce the radius, the point p that comes from the seed, the queried source S and the locations L_p^r. The stopping condition is retrieving less than M_S locations (line 3). However, we have no control over the number of additional requests needed by *MSSD-R*, but under the following assumption, we know that *MSSD-R* converges:

Assumption A.2. *Let S be a source and $L(S)$ its locations. For each point p there exists a radius r_p such that the surface covered by Circle(p, r_p) contains less than M_S locations.*

We assume that there will always be an r_p such that $|L_p^{r_p}| < M_S$. *MSSD-R* performs several requests decreasing r by α until r_p is found and *MSSD-R* reaches the stopping condition. Given Assumption A.2, we guarantee that *MSSD-R* performs a finite number of requests.

Algorithm A.1c *MSSD* Recursive (*MSSD-R*)

```
6: for each S in {S_1, S_2, ...S_k} - S_seed do
7:    L_p^r ← ∅
8:    L_p^r ← RadRecursive(r_r, α, p, S, L_p^r)
9:    L_S ← L_S ∪ L_p^r
10: end for
```

Algorithm A.2 RadRecursive

Input: r_r, α, p, S, L_p^r
Output: L_p^r

```
1: R ← API(p, r_r, S)          →/* Query S with r_r*/
2: L_p^r ← L_p^r ⋃ R
3: if |R| < M_S then
4:    return L_p^r             →/* The area is not dense*/
5: else
6:    RadRecursive(r_r/α, α, p, S, L_p^r)          →/* Call with new r_r*/
7: end if
```

5.3 Optimizing the Point Selection

All *MSSD* algorithms are based on exhaustive querying. However, some seed points might be quite close to each other, resulting in redundant API requests. We propose *MSSD** which clusters the seed points using DBSCAN [39] (a clustering algorithm suitable for spatial data and robust to noise) and queries with the centroids of the clusters. If the results size is maximal, then there is a possibility that this is a dense area. Afterward, we apply DBSCAN on *the union of the points of the current cluster and the points retrieved from the API request*. Depending on the data distribution of the source, we move the focus to the dense areas that we discover. We stop when the result size of the request is less than the maximal (based on Theorem A.1).

Algorithm A.2a RadRecursive*

Input: r_r, α, $< p, C_p >$, S, ϵ, m, L_p^r
Output: L_p^r
6: $\{C\}' \leftarrow DBSCAN(C_p \bigcup R, \frac{\epsilon}{\alpha}, \frac{m}{\alpha})$ →/* DBSCAN on the union of C_p and R with new parameters*/
7: **for each** $< c', C' >$ **do**
8: RadRecursive*($\frac{r_r}{\alpha}$, α, $< c', C'_c >$, S, $\frac{\epsilon}{\alpha}$, $\frac{m}{\alpha}$, L_p^r)
9: **end for**

Fig. A.4 shows a simple example of *MSSD**. The seed points come from the seed, whereas the blue ones are in the source. The initial DBSCAN will cluster together (A, B, C), (D, E), (F,G, H) and I. After the querying with the centroids of these clusters, only clusters (A, B, C) and I will continue further. The new clusters for (A, B, C) will be A, B and (C, K, L), where K and L are points from the source. For cluster I, we query with the centroid of (I, J, M). In the third step cluster (I, J, M) is divided to I, (J,M) and N, where N is a new point discovered from the second step. *MSSD** is formalized in Alg. A.3. After a source is chosen, its points are clustered with DBSCAN (line 5) using ϵ as minimal distance between points and m as the number of points that a cluster should have. DBSCAN returns the set of clusters $\{C\}$. For each centroid c of the cluster C, we call RadRecursive* (Alg. A.2a), which is similar to its parent version, RadRecursive (Alg. A.2) regarding the stopping condition and the adaptive radius but differs from line 6 and on (the *else* clause). If the area is dense, then we split the cluster by taking into consideration the union $C_p \bigcup R$ of points in the cluster C_p and the retrieved points from the source R. We cluster $C_p \bigcup R$ with DBSCAN in line 6 and we receive a new set of clusters $\{C'\}$. For each centroid c' of the cluster C' we call the algorithm again with the adjusted parameters. Note that in the case of Twitter, the majority of results R is polygons. Therefore we modify line 6 in Alg. A.2a with (i) the centroids of the polygons and we denote this version

of *MSSD*-C* or (ii) the nearest point of the polygon to the queried point p, and we denote this version as *MSSD*-N*.

Algorithm A.3 *MSSD** algorithm

Input: A set of sources $\{S_1, S_2, ...S_n\}$, radius r
Output: $\{L^*_{S_1}, L^*_{S_2}, ...L^*_{S_k}\}$

1: **for each** S in $\{S_1, S_2, ...S_k\}$ - S_{seed} **do**
2: $L_S \leftarrow L_I = \bigcup_{i=1}^{k} L_i$ $\rightarrow$/* Initialize each L_S with L_I*/
3: **end for**
4: Let S_{seed} be the source with the most points in $\{S_1, S_2, ...S_k\}$, L_{seed} its locations and P the distinct points in L_{seed}
5: $\{C\} \leftarrow DBSCAN(P), \epsilon, m)$
6: **for each** S **do**
7: **for each** $< c, C >$ **do**
8: $L^r_p \leftarrow \emptyset$
9: $L^r_p \leftarrow$ RadRecursive*$(r, \alpha, < c, C_c >, S, \epsilon, m, L^r_p)$
10: $L_S \leftarrow L_S \cup L^r_p$
11: **end for**
12: **end for**
return $\{L^*_{S_1}, L^*_{S_2}, ...L^*_{S_k}\}$

*MSSD** has these advantages: (i) *MSSD** manages better the requests by using the centroids of clusters rather than all the points in a cluster, (ii) *MSSD** is not sensitive to parameters because it uses an adaptive algorithm to learn them for each of the sources, and (iii) while querying, *MSSD** adapts the center of the circle depending on the locations found by the previous query. Let us now suppose that the optimal combination of pairs of $< p^*, r^* >$ that retrieve the maximal L^* exists. In order to compare our solution to the optimal, let us first prove the submodularity of our problem.

Theorem A.2. *Optimizing the data extraction based on API calls is a monotone submodular problem.*

Proof. An API call takes $< p, r >$ as parameters and retrieves L^r_p locations. Let us denote as $\gamma(p, r)$ the *gain* (*new locations*) that $API(p, r)$ brings. Note that an extra API call is effective as long as it contributes to the union of the results of the previous calls. To prove the *submodularity*, we need to show that $\gamma(P' \cup p, r) \geq \gamma(P \cup p, r)$ if $P' \subset P$. Let us consider a set of points P and P' such that $P' \subset P$. The locations retrieved from P' are $\bigcup_{i=1}^{|P'|} L^r_{p_i}$ and the locations retrieved from P are $\bigcup_{i=1}^{|P|} L^r_{p_i}$. Since $P' \subset P$, $\bigcup_{i=1}^{|P'|} L^r_{p_i} \subseteq \bigcup_{i=1}^{|P|} L^r_{p_i}$. Let us consider a new point p. and L^r_p the result of $API(p, r)$.

Since $\bigcup_{i=1}^{|P'|} L_{p_i}^r \subseteq \bigcup_{i=1}^{|P|} L_{p_i}^r$, then $(L_p^r \cap (\bigcup_{i=1}^{|P'|} L_{p_i}^r)) \subseteq (L_p^r \cap (\bigcup_{i=1}^{|P|} L_{p_i}^r))$. As a result, $\gamma(P' \cup p, r) \geq \gamma(P \cup p, r)$. To prove the *monotonicity*, for every $P' \subseteq P$, $|\bigcup_{i=1}^{|P'|} L_{p_i}^r| \leq |\bigcup_{i=1}^{|P|} L_{p_i}^r|$. So, the more we increase the set of seed points, the more locations we get. It is simple to show that $\bigcup_{i=1}^{|P|} L_{p_i}^r = (\bigcup_{i=1}^{|P'|} L_{p_i}^r) \cup (\bigcup_{i-i}^{|P-P'|} L_{p_i}^r)$ so $(\bigcup_{i=1}^{|P'|} L_{p_i}^r) \cup (\bigcup_{i-i}^{|P-P'|} L_{p_i}^r) \supseteq \bigcup_{i=1}^{|P'|} L_{p_i}^r$. Hence, $|\bigcup_{i=1}^{|P'|} L_{p_i}^r| \leq |\bigcup_{i=1}^{|P|}$. □

Our *MSSD** tries to solve the data extraction problem by providing a solution that starts with initial centroids and then splits further if the area looks promising in terms of density. However, we extract only M_S locations in one call, and this sample might not be representative if the amount of the actual locations in the area may be quite large. So, if the sample of the M_S points misses some dense areas, our DBSCAN will classify those as outliers, and we will not query further. Thus, our solution is *greedy* because it makes a locally optimal solution regarding which API calls to perform in step $i+1$ based on the information of step i.

Theorem A.3. *The greedy solution MSSD* of our monotone submodular problem performs at least* $1 - \frac{1}{e}$ *as good as the optimal solution in terms of maximizing the number of locations, where e is the base of the natural logarithm.*

A greedy approach to a monotone submodular problem is guaranteed to be at least $1 - \frac{1}{e}$ as good as the optimal solution [40]. The proof uses the submodularity and the monotonicity to show the ratio between the greedy and the optimal solution.

Proof. Let L^* be the result of the optimal solution from points P^* and L_k the greedy solution provided by *MSSD** for n requests. Note that L^* is not the same as the total locations $L(S)$ of the source S but instead the optimal solution given P^* starting seed points and obeying the limitations of the API. Due to the monotonicity, we can write: $\bigcup_{i=1}^{|P^*|} L_{p_i}^r \leq \bigcup_{i=1}^{|P^* \cup P'|} L_{p_i}^r = \bigcup_{i=1}^{|P'|} L_{p_i}^r + \sum_{j=1}^{n} \gamma(p_j, r) \leq \bigcup_{i=1}^{|P'|} L_{p_i}^r + n(\bigcup_{i=1}^{|P'+1|} L_{p_i}^r - \bigcup_{i=1}^{|P'|} L_{p_i}^r)$ and $\bigcup_{i=1}^{|P^*|} L_{p_i}^r - \bigcup_{i=1}^{|P'|} L_{p_i}^r \leq n(\bigcup_{i=1}^{|P'+1|} L_{p_i}^r - \bigcup_{i=1}^{|P'|} L_{p_i}^r)$. We rearrange as: $\bigcup_{i=1}^{|P^*|} L_{p_i}^r - \bigcup_{i=1}^{|P'|} L_{p_i}^r \leq n((\bigcup_{i=1}^{|P^*|} L_{p_i}^r - \bigcup_{i=1}^{|P'|} L_{p_i}^r) - (\bigcup_{i=1}^{|P^*|} L_{p_i}^r - \bigcup_{i=1}^{|P'+1|} L_{p_i}^r))$ and we use δ_i to represent $\bigcup_{i=1}^{|P^*|} L_{p_i}^r - \bigcup_{i=1}^{|P'|} L_{p_i}^r$ so we can rewrite: $\delta_i \leq n(\delta_i - \delta_{i+1})$ and finally $\delta_{i+1} \leq (1 - \frac{1}{k})\delta_i$. So, for every $k \leq n$ we can write $\delta_k \leq (1 - \frac{1}{n})^k \delta_0$. Note that $\delta_0 = \bigcup_{i=1}^{|P^*|} L_{p_i}^r - \bigcup_{i=1}^{|\emptyset|} L_{p_i}^r = \bigcup_{i=1}^{|P^*|} L_{p_i}^r$. Moreover, for all $x \in R$, $1 - x \leq e^{-x}$. So finally, we can write that $\delta_k \leq (1 - \frac{1}{n})^k \bigcup_{i=1}^{|P^*|} L_{p_i}^r \leq e^{-\frac{k}{n}} \bigcup_{i=1}^{|P^*|} L_{p_i}^r$. By substituting δ_k with $\bigcup_{i=1}^{|P^*|} L_{p_i}^r - \bigcup_{i=1}^{|P^k|} L_{p_i}^r$, rearranging and finally replacing $\bigcup_{i=1}^{|P^k|} L_{p_i}^r$

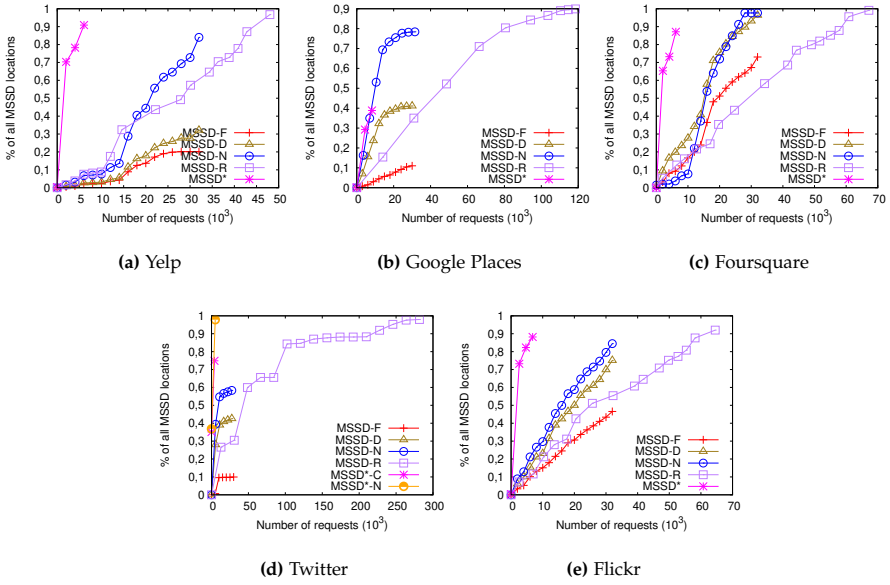
with its result L_k and $\bigcup_{i=1}^{|P^*|} L_{p_i}^r$ with L^*, we have: $L_k \geq (1 - e^{-\frac{k}{n}})L^*$ and for $l = k$ (lower bound) we have: $L_k \geq (1 - \frac{1}{e})L^*$. □

6 Experiments

In this section, we test our approach on the sources presented in Section 4 and compare with the existing baselines.

6.1 MSSD Experiments

We run *MSSD* algorithms using Krak as the seed source as it is the richest in terms of locations, points, and categories. We compare the results of the baseline (the initial locations of sources *SI*) to *MSSD-F* which uses a fixed radius of 2 km (Alg. A.1), *MSSD-D* with a density-based approach to define the radius (Alg. A.1a), *MSSD-N* with a nearest-neighbor method to define a flexible radius (Alg. A.1b) and *MSSD-R* with a recursive method that starts with a radius of 16 km (the largest values accepted by all sources) and reduces the radius by a coefficient $\alpha = 2$ (Alg. A.1c). Some APIs allow only an integer radius in the granularity of km, so $\alpha = 2$ is the smallest integer value accepted. Fig. A.6 illustrates the improvement in the extracted data volume from each source by each version of *MSSD* over *SI*. Krak is not included since it is the *seed*. Google Places (GP) has the highest improvement of 98.4 times more locations extracted by *MSSD-R* compared to the initial ones from *SI*. Flickr had 4,084 locations initially, which become 4.3 times more with *MSSD-F* and above 9.5 times more with *MSSD-D*, *MSSD-N* and *MSSD-R*. In Foursquare (FSQ) and Yelp, *MSSD-F* extracts 3 and 2 times more locations respectively, but *MSSD-D*, *MSSD-N*, and *MSSD-R* retrieve up to 3.5. Twitter returns 10.7 times more with *MSSD-R* but still has a low number of locations overall. These values highlight that *in spite of their different scopes, all the sources relate to the same physical world*. *MSSD-R* performs the best with an improvement of 14.3 times more than *SI* but with extra requests that in the case of Twitter and GP can reach up to 8 times more than *MSSD-F*, *MSSD-D* and *MSSD-N*. We ran the optimized version *MSSD** (Alg. A.3) for each of the sources with initial radius of 16 km and initial $m = 10$ and $\epsilon = 500$ meters as parameters of DBSCAN. m, ϵ and r are recursively reduced by $\alpha = 2$. We compared *MSSD-F*, *MSSD-D*, *MSSD-N*, *MSSD-R* and *MSSD** regarding the number of requests performed and the locations retrieved. The results for each source are presented in Fig. A.5. The number of requests is in the x-axis, whereas the number of locations is expressed as the percentage of the total of distinct locations extracted by all methods. *MSSD-R* provides the highest percentage of locations (above 95%) for all the sources but considerably more requests. For instance, for GP (Fig. A.5b) and for Twitter (Fig. A.5d), *MSSD-R* need re-



(a) Yelp **(b)** Google Places **(c)** Foursquare

(d) Twitter **(e)** Flickr

Fig. A.5: Requests versus locations for different *MSSD* algorithms with Krak as seed

spectively 3.8 and 8.7 times more requests than the *MSSD* versions with fixed number of requests (*MSSD-F*, *MSSD-D*, *MSSD-N*). For the same number of requests, *MSSD-N* provides a higher percentage of locations compared to *MSSD-F* and *MSSD-D* for all the sources. *MSSD** is the most efficient in terms of requests. For all sources except Google Places, *MSSD** gets around 90% of the locations with around 25% of the requests of *MSSD-F*, *MSSD-D* and *MSSD-N*. With regard to *MSSD-R*, *MSSD** uses 12%-15% of *MSSD-R* requests for Flickr, Yelp and Foursquare, 8.5% of *MSSD-R* requests for Google Places and 2.7% of *MSSD-R* requests for Twitter. In Google Places, *MSSD** can retrieve only 40% of the locations, because of its small result size of Google (Table A.1). *MSSD-N* extracts 2 times more locations than *MSSD** but with 3 times more requests. In Twitter, *MSSD*-C* (with centroids) retrieves 20% more locations than *MSSD*-N* (with the nearest neighbor) using the same number of requests. To conclude, *MSSD** guarantees the best compromise for all the sources.

Setting α and radius r. In this experiment, we test different values of α and r for *MSSD**. When α is bigger, or r is smaller, fewer requests are performed, some areas are missed, and consequently, fewer locations are retrieved. Table A.3 provides the trade-offs in terms of percentage of requests and percentage of locations of *MSSD** with regards to *MSSD-R* for each α (while fixing the radius at 16 km) and for each r (while fixing α at 2)(See Appendix B.2 in

Sources	Req vs loc	Alpha								Radius				
		2	4	6	8	10	12	14	16	1	4	8	12	16
FSQ	% req	13.2%	13.0%	12.9%	12.9%	12.9%	12.9%	12.9%	12.9%	12.8%	12.9%	13.1%	13.3%	13.2%
	% loc	88.3%	78.1%	75.4%	74.8%	73.0%	71.5%	70.3%	69.5%	43.2%	82.9%	86.4%	88.1%	88.3%
Flickr	% req	15.8%	15.4%	15.2%	15.1%	15.0%	15.0%	15.0%	15.0%	13.2%	14.4%	15.3%	15.7%	15.8%
	% loc	96.5%	91.9%	86.9%	85.1%	82.9%	81.4%	80.3%	79.0%	49.1%	88.5%	94.8%	95.8%	96.5%
GP	% req	9.3%	9.0%	8.8%	8.7%	8.6%	8.5%	8.5%	8.4%	7.6%	9.0%	9.3%	9.3%	9.3%
	% loc	38.4%	34.3%	32.9%	32.4%	31.6%	30.9%	30.4%	30.1%	33.6%	37.2%	37.3%	38.2%	38.4%
Yelp	% req	17.5%	17.4%	17.4%	17.4%	17.4%	17.4%	17.4%	17.4%	17.3%	17.4%	17.4%	17.4%	17.5%
	% loc	98.2%	95.8%	93.7%	95.1%	93.4%	90.6%	89.8%	89.7%	51.1%	94.2%	97.7%	98.3%	98.2%
Twitter-C	% req	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%
	% loc	76.8%	58.6%	58.1%	57.1%	55.7%	52.8%	52.3%	52.3%	53.4%	61.5%	60.0%	58.9%	76.8%
Twitter-N	% req	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%	3.0%
	% loc	99.5%	99.4%	99.0%	98.6%	98.3%	98.2%	97.9%	97.8%	86.4%	97.0%	97.1%	98.4%	99.5%

Table A.3: % of req. vs % of loc. for *MSSD** relative to *MSSD-R* depending on α and r

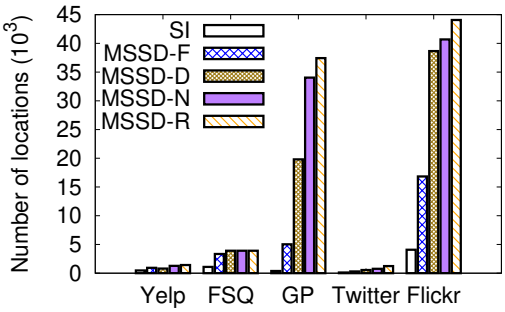
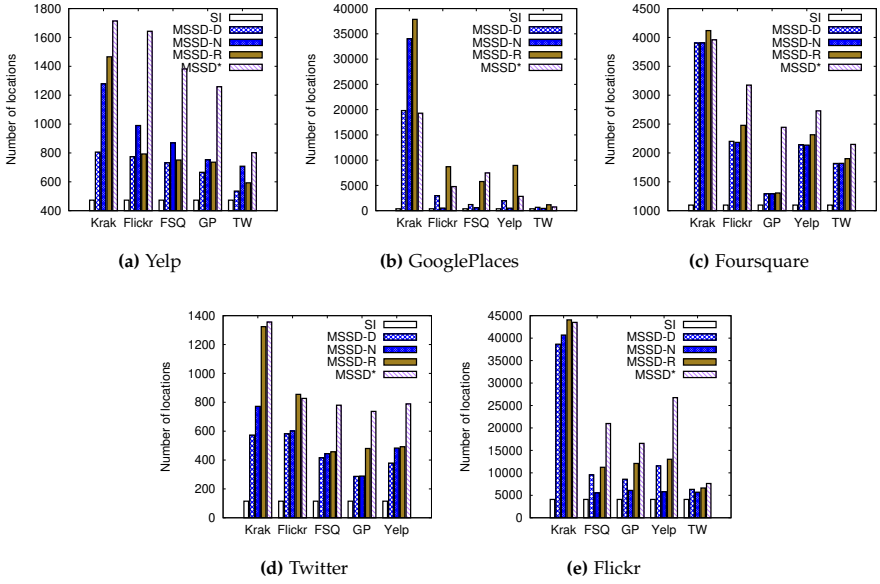


Fig. A.6: Nr of loc. per source

the full version of the paper [4] for details.). In all the cases, the additional requests of *MSSD** with small values of α are rewarded with a higher percentage of locations. For example, for 0.3% more requests, we retrieve 18.8% more locations in Foursquare. In Flickr, for 0.8% more requests, we retrieve 17.5% more locations. Similarly, starting with a big radius is safer and more rewarding. For instance, Foursquare and Yelp perform less than 0.4% of the requests to get around 46% more locations when starting with r=16km compared to r=1km. *A risk-averted selection of parameters turns out to provide a good trade-off between the number of requests and number of locations because MSSD* adapts to the density of the region and still manages the requests carefully*. Thus, the algorithm is robust to different parameter settings, and fine-tuning is not needed.

Choosing a different seed. In order to show that our *MSSD* algorithms apply to any type of seed (preferable a rich source), we ran *MSSD-D*, *MSSD-N*, *MSSD-R* and *MSSD**using as seed Flickr, Foursquare, Yelp, Google Places, and Twitter. The results are presented in Fig. A.7. Even though Krak performs the best, the other sources which provide significantly fewer seed



(a) Yelp **(b)** GooglePlaces **(c)** Foursquare

(d) Twitter **(e)** Flickr

Fig. A.7: *MSSD* results with different seeds for all sources

points (see Table A.2) are able to achieve comparable results. Recall that Krak has 14 times more seed points than Flickr, 30 times more than Foursquare, 70 times more than Yelp, 91 times more than Google Places and 295 times more than Twitter. Apart from Krak, *MSSD** performs the best for Flickr, Yelp, and Foursquare. For Flickr, *MSSD** with Yelp seed points retrieves 6.5 times more points than *SI*. For FSQ, *MSSD** with Flickr and Yelp seed points retrieves 2.9 and 2.5 times more points than *SI*, respectively. For Yelp, *MSSD** with Flickr seed points retrieves 3.5 times more data than *SI*, whereas Krak retrieves 3.6 times more while having 70 times more seed points. For Twitter and Google Places, *MSSD-R* performs the best; 7.4 times more locations than *SI* with Flickr seed points in Twitter and 23.6 times more locations with Yelp seed points in Google Places. The performance of each algorithm in terms of the number of requests versus the number of locations can be found in [4]. An interesting observation is that *MSSD** sometimes performs better than *MSSD-R*. *Even when the seed source is not rich, MSSD* manages to achieve good results due to its ability to adapt the next call according to the distribution of the source.*

Elapsed time of the experiments. The elapsed time is more important than the CPU time because of the *bandwidth limitations*. Our experiments were run simultaneously for all the sources in order to avoid the temporal bias, so all bandwidth limitations were respected at the same time. The elapsed time

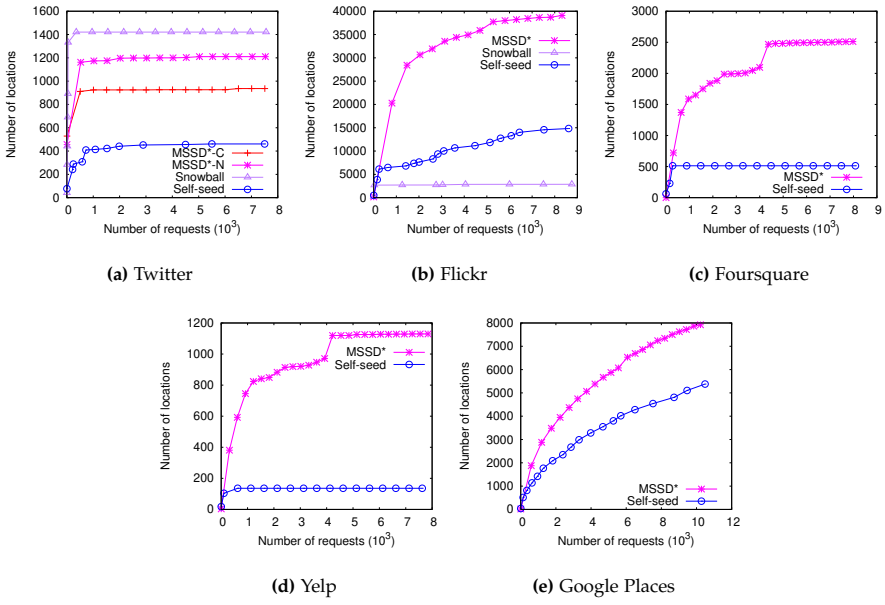


Fig. A.8: Number of request versus number of locations for *MSSD**, *Snowball* and *Self-seed*

for all the sources is around 1 week for the *MSSD-F*, *MSSD-D* and *MSSD-N*, 2 weeks for *MSSD-R* and 1.7 days for *MSSD**. If the algorithms are run independently, it takes on average 1 day per source for *MSSD-F*, *MSSD-D* and *MSSD-N*, 2 days for *MSSD-R* and less than 6 hours for *MSSD**.

6.2 Comparison with Existing Baselines

The technique using *linked accounts* [33–35] requires users that have declared their account in another social network. From our initial querying of the sources, there were only 0.27 % of users on Flickr with linked accounts to Twitter and 0.003 % of users on Twitter with linked accounts to Foursquare. Hence, a comparison with this technique makes little sense. The *keyword-based* querying shows limited applicability in location-based data retrieval. We conducted a small experiment using the names of cities and towns in North Denmark as keywords. For Flickr and Twitter, the precision (% of data that falls in the queried region) was just 31.6% and 0.85% respectively, while the recall was less than 5%, relative to *MSSD**. Foursquare and Yelp offer a query by term or query by location expressed as a string. The former [query by term] does not retrieve any data when queried with a city or town name. If we express the location as a string, the precision is 93% and 85% for Foursquare and Yelp respectively, and the recall is less than 19%, relative to

Source	Algorithm	Real 80%	Synthetic 20%	Req	Real 70%	Synthetic 30%	Req	Real 50%	Synthetic 50%	Req
Flickr	*MSSD-R*	85.62%	87.16%		84.44%	85.70%		82.71%	83.70%	
	*MSSD**	64.91%	66.22%	10.16%	63.59%	65.01%	10.12%	62.12%	64.28%	9.80%
Yelp	*MSSD-R*	97.17%	99.44%		97.10%	99.4%		97.10%	99.23%	
	*MSSD**	72.23%	77.81%	12.74%	68.89%	76.94%	12.49%	66.42%	72.59%	11.78%
FSQ	*MSSD-R*	94.27%	96.69%		94.19%	95.51%		94.07%	95.26%	
	*MSSD**	67,75%	74,57%	10.67%	66,15%	70,34%	10.62%	63,60%	68,65%	10.45%
GP	*MSSD-R*	92,42 %	78,76%		91,60%	76,15%		89,94%	69,33%	
	*MSSD**	34,48%	35,76%	13,85%	45,80%	33,59%	9,14%	38,09%	33,99%	13,49%
Twitter	*MSSD-R*	81,16 %	97,70%		81,16%	95,66%		80,92%	87,23%	
	MSSD-C*	45,30%	63,31%	2,45%	44,37%	62,09%	2,44%	48,88%	68,51%	2,45%
	MSSD-N*	69,79%	97,70%	2,53%	69,11%	94,51%	2,53%	60,90%	86,61%	2,53%

Table A.4: *MSSD-R* and *MSSD** compared to ground truth

*MSSD**. In GP, the data retrieved is only the towns and the cities themselves. For example, if we query with the keyword "Aalborg", the API will return Aalborg city only and not any other places located in Aalborg (1 request per 1 location). Even though the precision is 100%, the recall is only 0.07% relative to *MSSD**. Thus, we compare to *Snowball* and to the technique mentioned in [36].

		Nr of loc.	Nr of users	Time period
Twitter	Snowball	1421	35	2015-2018
	Self-seed	461	101	2017-2018
	MSSD*-C	936	195	2017-2018
	MSSD*-N	1237	231	2017-2018
Flickr	Snowball	2885	46	2005-2018
	Self-seed	14910	1007	2005-2018
	MSSD*	39427	1740	2005-2018

Table A.5: *Snowball*, *Self-seed* and *MSSD** comparison

To compare with *Snowball* ([30, 31]), we formed the seed with the users found in Section 4. We used the same number of requests for *Snowball* and *MSSD**. *Snowball* is based on users, and consequently, it can be applied only to Twitter and Flickr (Foursquare API no longer provides the check-in data unless the crawling user has checked in himself at the venue). The technique mentioned in [36] (we will refer to it as *Self-seed*) starts with querying a specific location to get initial points. Later, other requests are performed using the seed points of the previous step. We ran *Self-seed* on all our sources for the same number of requests as *MSSD** (results in Fig. A.8). *Snowball* in Twitter retrieves more locations in the region than versions of *MSSD** (*MSSD*-C* and *MSSD*-N*) and *Self-seed* (Fig. A.8a) because in the case of user-based calls, the bandwidth is 200 (100 for location-based) and the historical access is unlimited (2 weeks for location-based). In the case of Flickr, *MSSD** outperforms *Snowball* and

Self-seed with 14 and 3 times more locations respectively. *Snowball* gets most of the data in the region in the beginning and degrades later because *when using Snowball, while we traverse the network, there is more and more data which falls outside the region of interest*. *MSSD** yields a higher number of locations compared to *Self-seed*: 5.5 times more locations for Foursquare, 9 times more locations for Yelp and 3.5 times more locations for Google Places.
Self-seed in the case of directories stops yielding new locations after approximately 500 requests. *In the case of directories, after some steps, the seed points in Self-seed stop growing, converging into a dead end*. Recall that Google Places has a result size of 20 and is denser in terms of data, so it has new locations for the subsequent steps, avoiding thus the convergence. The number of users and the time period covered are presented in Table A.5. Despite the slight advantage of *Snowball* on Twitter in terms of the number of locations, the data comes only from 35 users compared to 231 for *MSSD*-N* and 101 for *Self-seed*. Moreover, the time period covered by the tweets in *Snowball* is 3 years compared to 1 year of *MSSD** versions. Regarding Flickr, the time period of the photos is the same, but the number of photos and the number of users are 1-2 orders of magnitude larger for *MSSD** compared to *Snowball*. *Self-seed* can retrieve a better variety of users and locations compared to *Snowball* but still contains only half the number of locations and users of *MSSD**.

6.3 MSSD-R and MSSD* Result Completeness

Given the API limitations, we cannot get the actual ground truth of source locations. Instead, we performed the following experiment: first, we union all the locations sets from all our algorithms (*SI*, *MSSD-F*, *MSSD-D*, *MSSD-N*, *MSSD-R* and *MSSD**) to create a dataset of real data; second, we learn the distribution $\mathcal{D}$ of the locations by dividing the area in a grid of 1km x 1km and assigning each grid cell d a probability $p_d \sim \mathcal{D}_d$; third, we generate synthetic locations in the area and assign them to a grid cell d with the estimated probability p_d. We consider the synthetic and the real data as ground truth. We implemented "simulated offline" API functions for each source, respecting the maximal result size for each of them. We ran our *MSSD-R* and *MSSD** on the ground truth data for different ratios of synthetic data as in Table A.4. The data retrieved by *MSSD-R* is above 94% of the ground truth in Yelp and Foursquare and above 80% of the ground truth in Flickr, Google Places, and Twitter. *MSSD** performs the best in Yelp and Twitter (*MSSD*-N*) with above 70% of the ground truth for all ratios of real versus synthetic data, followed by Foursquare and Flickr with above 64%. What is more important, *MSSD-R and MSSD* are seen to be robust regardless of the ratio of synthetic to real data*. Although *MSSD** retrieves less than *MSSD-R*, this result is achieved using only around 10% of the requests of *MSSD-R*. In the case of Google Places, *MSSD** gets around 40% of the ground truth because of the small result size

of only 20 locations per request. In the case of Twitter, *MSSD*-N* performs better than *MSSD*-C*.

6.4 Discussion of Experiments

Selecting an external seed of points improved the number of locations retrieved and avoid converging into a dead end like in *Self-seed*. Moreover, the attempts to adapt the radius of the search according to the search region prove to be effective in retrieving more locations. *MSSD-F*, *MSSD-D* and *MSSD-N* extract on average up to 11.1 times more data than *SI* but if we adapt the radius according to the source (*MSSD-R*), we extract up to 14.3 times more locations than *SI*. *MSSD** provides a very good trade-off between the number of requests and number of locations as *MSSD** extracts up to 90% of the data of *MSSD-R* with less than 16% of its requests. Our comparison with the *Snowball* and the *Self-seed* baseline shows that our seed-driven algorithm is better in terms of extracting (i) up to 14 times more locations for all the sources, (ii) in the case of Twitter and Flickr, the activity originates from *a larger base of users* (up to 6.6 times more), and (iii) in the case of directories, our *MSSD* avoids converging into a dead end. In a ground truth dataset, for most of the sources, our *MSSD-R* algorithm finds 82 % - 99 % of the ground truth, while *MSSD** with 10% of the requests is able to guarantee 63 % - 73% of the ground truth.

7 Conclusions and future work

This paper was motivated by the need for an efficient algorithm that extracts recent geo-social data. We formulated the problem of data extraction as an optimization problem which aims to maximize the retrieved locations while minimizing the requests. We identified the API limitations for six sources: Krak, Yelp, Google Places, Foursquare, Twitter, and Flickr. Then, we proposed a seed-driven algorithm that uses one source as the *seed* to feed the points as API parameters to the others. *MSSD* versions extracted up to 14.3 times more data than *SI*. Our optimized algorithm *MSSD** retrieved 90% of the locations with less than 16% of the requests, outperforming *MSSD-D* and *MSSD-N*. Interesting directions for future research include applying machine learning for data extraction, seed selection based on other criteria (diversity in semantics, maximal spread of points, relation to the source), data integration, and data fusion of location-based data from multiple geo-social sources.

References

[1] Z. Cheng, J. Caverlee, and K. Lee, "You are where you tweet: a content-based approach to geo-locating twitter users," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, 2010, pp. 759–768.

[2] D. Jurgens, "That's what friends are for: Inferring location in online social media platforms based on social relationships," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 7, no. 1, 2013.

[3] L. Li, M. F. Goodchild, and B. Xu, "Spatial, temporal, and socioeconomic patterns in the use of twitter and flickr," *Cartography and geographic information science*, vol. 40, no. 2, pp. 61–77, 2013.

[4] S. Isaj and T. B. Pedersen, "Seed-driven geo-social data extraction–full version," *arXiv preprint arXiv:1901.06712*, 2019.

[5] H. Wang, M. Terrovitis, and N. Mamoulis, "Location recommendation in location-based social networks using user check-in data," in *Proceedings of the 21st ACM SIGSPATIAL international conference on advances in geographic information systems*, 2013, pp. 374–383.

[6] T. Emrich, M. Franzke, N. Mamoulis, M. Renz, and A. Züfle, "Geo-social skyline queries," in *International Conference on Database Systems for Advanced Applications*. Springer, 2014, pp. 77–91.

[7] G. Li, S. Chen, J. Feng, K.-l. Tan, and W.-S. Li, "Efficient location-aware influence maximization," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 87–98.

[8] J. L. Z. Cai, M. Yan, and Y. Li, "Using crowdsourced data in location-based social networks to explore influence maximization," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[9] M. A. Saleem, R. Kumar, T. Calders, X. Xie, and T. B. Pedersen, "Location influence in location-based social networks," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 2017, pp. 621–630.

[10] F. Yu, N. Che, Z. Li, K. Li, and S. Jiang, "Friend recommendation considering preference coverage in location-based social networks," in *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 2017, pp. 91–105.

[11] J. Li, T. Sellis, J. S. Culpepper, Z. He, C. Liu, and J. Wang, "Geo-social influence spanning maximization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 8, pp. 1653–1666, 2017.

[12] Q. Zhu, H. Hu, C. Xu, J. Xu, and W.-C. Lee, "Geo-social group queries with minimum acquaintance constraints," *The VLDB Journal*, vol. 26, no. 5, pp. 709–727, 2017.

[13] J.-D. Zhang and C.-Y. Chow, "igslr: personalized geo-social location recommendation: a kernel density estimation approach," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2013, pp. 334–343.

[14] G. Ference, M. Ye, and W.-C. Lee, "Location recommendation for out-of-town users in location-based social networks," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013, pp. 721–726.

[15] M. A. Saleem, R. Kumar, T. Calders, and T. B. Pedersen, "Effective and efficient location influence mining in location-based social networks," *Knowledge and Information Systems*, vol. 61, no. 1, pp. 327–362, 2019.

[16] S. Feng, X. Li, Y. Zeng, G. Cong, and Y. M. Chee, "Personalized ranking metric embedding for next new poi recommendation," in *IJCAI'15 Proceedings of the 24th International Conference on Artificial Intelligence*. ACM, 2015, pp. 2069–2075.

[17] Z. Yao, Y. Fu, B. Liu, Y. Liu, and H. Xiong, "Poi recommendation: A temporal matching between poi popularity and user regularity," in *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE, 2016, pp. 549–558.

[18] S. Zhao, T. Zhao, H. Yang, M. Lyu, and I. King, "Stellar: Spatial-temporal latent ranking for successive point-of-interest recommendation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[19] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 1082–1090.

[20] H. Gao, J. Tang, X. Hu, and H. Liu, "Exploring temporal effects for location recommendation on location-based social networks," in *Proceedings of the 7th ACM conference on Recommender systems*, 2013, pp. 93–100.

[21] D. Jurgens, T. Finethy, J. McCorriston, Y. Xu, and D. Ruths, "Geolocation prediction in twitter using social networks: A critical analysis and

review of current practice," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 9, no. 1, 2015.

[22] N. Li and G. Chen, "Analysis of a location-based social network," in *2009 international conference on computational science and engineering*, vol. 4. Ieee, 2009, pp. 263–270.

[23] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo, "A random walk around the city: New venue recommendation in location-based social networks," in *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*. Ieee, 2012, pp. 144–153.

[24] C. Zhang, L. Shou, K. Chen, G. Chen, and Y. Bei, "Evaluating geo-social influence in location-based social networks," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012, pp. 1442–1451.

[25] M. Weiler, K. A. Schmid, N. Mamoulis, and M. Renz, "Geo-social co-location mining," in *Second International ACM Workshop on Managing and Mining Enriched Geo-Spatial Data*, 2015, pp. 19–24.

[26] S. Scellato, A. Noulas, and C. Mascolo, "Exploiting place features in link prediction on location-based social networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 1046–1054.

[27] F. Burini, N. Cortesi, K. Gotti, and G. Psaila, "The urban nexus approach for analyzing mobility in the smart city: Towards the identification of city users networking," *Mobile Information Systems*, vol. 2018, 2018.

[28] A. Likhyani, S. Bedathur, and P. Deepak, "Locate: Influence quantification for location promotion in location-based social networks." in *IJCAI*, 2017, pp. 2259–2265.

[29] Y. Liu, T.-A. N. Pham, G. Cong, and Q. Yuan, "An experimental evaluation of point-of-interest recommendation in location-based social networks," *Proceedings of the VLDB Endowment*, vol. 10, no. 10, pp. 1010–1021, 2017.

[30] H. Gao, J. Tang, X. Hu, and H. Liu, "Content-aware point of interest recommendation on location-based social networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 29, no. 1, 2015.

[31] S. Scellato, C. Mascolo, M. Musolesi, and V. Latora, "Distance matters: Geo-social metrics for online social networks." in *WOSN*, 2010.

[32] S. H. Lee, P.-J. Kim, and H. Jeong, "Statistical properties of sampled networks," *Physical review E*, vol. 73, no. 1, p. 016102, 2006.

[33] D. Hristova, M. J. Williams, M. Musolesi, P. Panzarasa, and C. Mascolo, "Measuring urban social diversity using interconnected geo-social networks," in *Proceedings of the 25th international conference on world wide web*, 2016, pp. 21–30.

[34] N. Armenatzoglou, S. Papadopoulos, and D. Papadias, "A general framework for geo-social query processing," *Proceedings of the VLDB Endowment*, vol. 6, no. 10, pp. 913–924, 2013.

[35] D. Preoţiuc-Pietro and T. Cohn, "Mining user behaviours: a study of check-in patterns in location based social networks," in *Proceedings of the 5th annual ACM web science conference*, 2013, pp. 306–315.

[36] R. Lee and K. Sumiya, "Measuring geographical regularities of crowd behaviors for twitter-based geo-social event detection," in *Proceedings of the 2nd ACM SIGSPATIAL international workshop on location based social networks*, 2010, pp. 1–10.

[37] N. Bennacer, F. Bugiotti, M. Hewasinghage, S. Isaj, and G. Quercini, "Interpreting reputation through frequent named entities in twitter," in *International Conference on Web Information Systems Engineering*. Springer, 2017, pp. 49–56.

[38] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 591–600.

[39] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[40] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Mathematical programming*, vol. 14, no. 1, pp. 265–294, 1978.

Paper B

Multi-Source Spatial Entity Linkage

Suela Isaj, Torben Bach Pedersen and Esteban Zimányi

The paper has been published in the
IEEE Transactions on Knowledge & Data Engineering (TKDE), April 27, 2020.

Abstract

*Besides the traditional cartographic data sources, spatial information can also be derived from location-based sources. However, even though different location-based sources refer to the same physical world, each one has only partial coverage of the spatial entities, describe them with different attributes, and sometimes provide contradicting information. Hence, we introduce the spatial entity linkage problem, which finds which pairs of spatial entities belong to the same physical spatial entity. Our proposed solution (*QuadSky*) starts with a time-efficient spatial blocking technique (*QuadFlex*), compares pairwise the spatial entities in the same block, ranks the pairs using Pareto optimality with the* SkyRank *algorithm, and finally, classifies the pairs with our novel* SkyEx-* *family of algorithms that yield 0.85* precision *and 0.85* recall *for a manually labeled dataset of 1,500 pairs and 0.87* precision *and 0.6* recall *for a semi-manually labeled dataset of 777,452 pairs. Moreover, we provide a theoretical guarantee and formalize the* SkyEx-FES *algorithm that explores only 27% of the skylines without any loss in* F-measure*. Furthermore, our fully unsupervised algorithm* SkyEx-D *approximates the optimal result with an* F-measure *loss of just 0.01. Finally,* QuadSky *provides the best trade-off between* precision *and* recall*, and the best* F-measure *compared to the existing baselines and clustering techniques, and approximates the results of supervised learning solutions.*

1 Introduction

Web data and social networks are growing in terms of information volume and heterogeneity. Almost all online sources offer the possibility to introduce locations (geo-tagged entities accompanied by semantic details). A specific type of sources whose primary focus is locations is *location-based sources*, such as Google Places, Yelp, Foursquare, etc. In contrast to cartographic data sources, locations in location-based sources have a hybrid form that stands between a *spatial object* and an *entity*. We refer to them as *spatial entities* since they are spatially located but also identified by other attributes such as the name of the location, the address, keywords, etc. Spatial entities play a key role in several systems that rely on spatial information such as geo-recommender systems, selecting influential locations, search engines using geo-preferences, etc.

However, while a spatial object is identified only by the coordinates, this is not the case for spatial entities. Different spatial entities might co-exist in the same coordinates (shops in a shopping mall), or the same entity might be located in different but nearby coordinates across different sources (e.g., "Chicago Roasthouse" appears in Yelp and Google Places with coordinates 82 meters apart). The identity of a spatial entity is the combination of several attributes. Unfortunately, the identity of a spatial entity is sometimes difficult to infer due to the inconsistencies within and among the sources; each location-based source contains different attributes; some attributes might be missing and even contradicting. For example, source A contains the spatial entity "Lygten" in (57.436 10.534) with the keywords "coffee", "tea", and "cocoa and spices", while source B contains "Restaurant Lygten" in (57.435 10.533) with the keyword "restaurant". We need a technique that can automatically decide whether these two spatial entities are the same real-world entity. The problem of finding which spatial entities belong to the same physical entity is referred to as *spatial entity linkage* or *spatial entity resolution*. We use the term entity linkage since we do not merge the entities [1].

There are several works that apply entity linkage in various fields [2–4, 4, 5, 5–10] but only little work on spatial entities [11–14], even though they are central in geo-related research. The entities in the majority of the entity linkage research refer to people; thus, the methodologies and the models are based on the similarities that two records of the same individual would reveal. Moreover, these works do not address the spatial character of spatial entities. As for the works in spatial entity integration [11–13], their main contribution is a tool rather than an algorithm. What is more, the methods propose arbitrarily attribute weights and score functions without experimentation nor evaluation. In contrast to [11–13], the skyline-based algorithm (*SkyEx*) proposed in [10] is free of scoring functions and semi-arbitrary weights, and achieves

good results. However, *SkyEx* is dependent on a threshold number of skylines k, which can only be discovered through experiments, as the authors do not provide methods for estimating k. To sum up, *on the one hand, there is a growing amount of information about spatial entities, both within a single source and across sources, which can improve the quality of the geo-information; on the other hand, the spatial entity linkage problem is hard to resolve not only because of the heterogeneity of the data but also because of the lack of appropriate and effective methods*.

In this paper, we address the problem of spatial entity linkage across different location-based sources. We significantly extend a previous conference paper [14]. As an overall solution building on [14], first, we propose a method that uses the geo-coordinates to arrange the spatial entities into blocks. Then, we pairwise compare the attributes of the spatial entities. Later, we rank the pairs according to their similarities using our novel technique, *SkyRank*. Finally, we introduce three approaches (*SkyEx-F*, *SkyEx-FES*, *SkyEx-D*) for deciding whether the pairs of compared entities belong to the same physical entity. Our contributions are: (1) we introduce *QuadSky*, a technique for linking spatial entities and we evaluate it on real-world data from four location-based sources; (2) we propose an algorithm called *QuadFlex* that organizes the spatial entities into blocks based on their spatial proximity, maintaining the complexity of a quadtree and avoiding assigning nearby points into different blocks; (3) to rank the pairs by their similarity, we propose a flexible technique (*SkyRank*) that is based on the concept of Pareto optimality; (4) to label the pairs, we propose the *SkyEx-** family of algorithms that considers the ranking order of the pairs and fixes a cut-off level to separate the classes; (5) we introduce two threshold-based algorithms: *SkyEx-F* that uses the *F-measure* to separate the classes, and *SkyEx-FES*, an optimized version of *SkyEx-F*, which provides a theoretical guarantee to prune 73% of the skyline explorations of *SkyEx-F*; (6) we propose *SkyEx-D*, a novel algorithm that is fully unsupervised and parameter-free to separate the classes.

Contributions 1 and 2 originate from [14], contributions 5 and 6 are new, and 3 and 4 are significantly improved compared to [14]. The work in [14] reported very good results compared to the baselines, but had the following limitation: the proposed threshold-based labeling algorithm *SkyEx* needed the threshold number of skylines k as input, and there were no proposed solutions on how to fix k, apart from experimenting with different values. We address this limitation by first modifying the original *SkyEx* in [14] as to only rank and not label the pairs, and we refer to it as *SkyRank*. Then, we delegate the classification problem to three new algorithms, namely *SkyEx-F*, *SkyEx-FES* and *SkyEx-D*. The experiments in [14] attempt to fix k using *precision*, *recall* and *F-measure*. We now formalize this rationale in our novel *SkyEx-F* algorithm. We improve further by providing a theoretical guarantee that *SkyEx-F* can be stopped before exploring the whole dataset, and propose the

optimized *SkyEx-FES* that prunes 80% of the skyline explorations of *SkyEx-F*. Furthermore, we introduce a novel approach for estimating the number of skylines (*SkyEx-D*), which is fully unsupervised and parameter-free and closely approximates the threshold-based versions (*SkyEx-F* and *SkyEx-FES*). In the present paper, we provide a new set of experiments for *SkyEx-FES* and *SkyEx-D*, and compare with *SkyEx-F*, supervised learning and clustering techniques.
The remainder of the paper is structured as follows: first, we describe the state of the art in Sect. 2; then, we introduce our approach in Sect. 3; later, we detail the stages of our approach: the spatial blocking in Sect. 4, comparing the pairs in Sect. 5, ranking the pairs in Sect. 6, and estimating the k^{th} level of skyline in Sect. 7; we analyze the complexity of our solution in Sect. 8; we provide experiments in Sect. 9; and finally, we conclude in Sect. 10.

2 Related Work

In this section, we describe some works on *entity resolution*, *spatial data integration*, and *spatial entity linkage*.
Entity resolution. The *entity resolution* problem has been referred in the literature with multiple terms including *deduplication*, *entity linkage*, and *entity matching* [4, 15]. Entity resolution has been used in various fields such as matching profiles in social networks [2], bioinformatics data [3], biomedical data [16], publication data [4, 5], genealogical data [6], product data [4, 5], etc. The attributes of the entities are compared, and a similarity value is assigned. The decision of whether to link two entities or not is usually based on a scoring function. However, finding an appropriate similarity function that combines the similarities of attributes and decides on whether to link or not the entities is often difficult. Several works use a training set to learn a classifier [7, 8, 17], others base the decision on a threshold derived through experiments [9, 18]. Other approaches decide the include the uncertainty of a match into the decision [19]. Finally, matching the entities can also be based on the feedback of an oracle [4, 5] or of a user [5].
Spatial data integration. There are several works on integrating purely spatial objects. Spatial objects differ from spatial entities mainly because a spatial object is fully determined by its coordinates or its spatial shape whereas a spatial entity, in addition to being geo-located, has a well-defined identity (name, phone, categories). The works on spatial object integration aim to create a unified spatial representation of the spatial objects from single/-multiple sources. Schafers at al [20] integrate road networks using rules for detect matching and non-matching roads based on the similarity in terms of the length, angles, shape, as well as the name of the street if available. The solutions in [21–24] are purely spatial and discuss the integration of spatial

objects originating from sensors and radars to have a better representation of the surface in 2D or even in 3D. These approaches cannot apply to spatial entities.

Spatial entity linkage. Accommodating the challenges of spatial entities for the entity resolution problem has been specifically addressed in [11–14, 25, 26]. The work in [25] is a bridge between the works in spatial data integration and spatial entity linkage because the entities have names, coordinates, and types but similarly to spatial objects, they refer to landscapes (rivers, deserts, mountains, etc.). The method used in [25] is supervised and requires labeled data. Moreover, even the similarity of the attribute "type" is learned through a training set. Regarding [11–13], the main contribution of these works relies on designing a spatial entity matching tool rather than an integration algorithm. In [13], the spatial entities within a radius are compared with each other, and the value of the radius is fixed depending on the type of spatial entity. For example, the radius is 50 m for restaurants and hotels, but 500 m for parks. All attributes (except coordinates) are compared using the Levenshtein similarity. Since the name, the geodata and the type of the entity are always present, they carry two-thirds of the weight in the scoring function whereas the weights of the website, the address and the phone number are tuned to one-third. The prototype of the spatial entity matching in [12] relies on a technique that arbitrarily uses an average of the similarity scores of all textual attributes without providing a discussing on this choice. Similarly to [11, 12], the main contribution of the work in [13] is designing a tool for spatial entity integration. The underlying algorithm considers spatial entities that are 5 m apart from each other and compares the name of the entities syntactically and the metadata related to an entity semantically. Finally, the decision is taken using the belief theory [26]. The works in [11–13] lack an evaluation of the algorithms. The work in [14] proposes a scalable spatial quadtree-based blocking technique that not only fixes the distance between the spatial entities but also controls the density of the blocks. Then, the spatial entities of the same block are compared on their name (Levenshtein), address (custom) and categories (Wu&Palmer using Wordnet). Finally, a threshold-based algorithm (*SkyEx*) is used to separate the classes. However, instead of using fixed thresholds for each attribute similarity, *SkyEx* abstracts the similarities into skylines and needs only one threshold number of skylines k to separate the classes. The authors provide experiments and evaluations, nevertheless, they lack estimation techniques for fixing k. The present paper uses the solution in [14] for the spatial blocking and the pairwise comparisons. We use the skylines for the labeling process as in *SkyEx*, but we propose three new algorithms (*SkyEx-F*, *SkyEx-FES* and *SkyEx-D*) to separate the classes, fixing k internally.

Summary. The general entity resolution approaches propose interesting solutions, but they do not consider the spatial character of a spatial entity. The

majority are designed to match entities that represent individuals (profiles in social networks, authors and publications, medical records, genealogical connections, etc.) or even linking species in nature. The proposed solutions for entity resolution in individuals, either supervised or based on an experimental threshold, are learned on human entity datasets. One can not merely assume the resemblance of behaviors in a human entity dataset to a spatial entity one. The solutions in species in nature are based on domain-specific algorithms that have little to no applicability in other fields. There is little specific work in spatial entities [11–13], mostly focusing on a tool for spatial data integration rather than on the algorithm. In all these works, the scoring function is chosen arbitrarily and no evaluation provided.

3 Spatial Entity Linkage

In this section, we introduce the problem definition and our overall solution. The basic concept used in this work is a *spatial entity* such as places, businesses, etc. Spatial entities originate from location-based sources, e.g., directories with location information (yellow pages, Google Places, etc.) and location-based social networks (Foursquare, Gowalla, etc.).

Definition B.1. *A spatial entity s is an entity identified uniquely within a source I, located in a geographical point p and accompanied by a set of attributes $A = \{a_i\}$.*

The attributes connected to s can be categorized as: *spatial*: the point where the entity is located, expressed in longitude and latitude; *textual*: attributes that are in the form of text such as name, address, website, description, etc.; *semantic*: attributes in the form of text that enrich the semantics behind a spatial entity, e.g., categories, keywords, metadata, etc.; *date, time or number*: other details about a spatial entity such as phone, opening hours, date of foundation, etc. An example of a spatial entity originating from Yelp can be a place named "Star Pizza" in the point (56.716 10.114), with the keywords "pizza, fast food", and with address "Storegade 31". The same spatial entity can be found again in Yelp or other sources, sometimes having the same attributes, more, less, or even attributes with contradictory values. Thus, there is a need for an approach that can unify the information within and across different sources in an intelligent manner.

Problem definition: *Given a set of spatial entities S originating from multiple sources, the spatial entity linkage problem aims to find those pairs of spatial entities $\langle s_i, s_j \rangle$ that refer to the same physical spatial entity.*

We propose *QuadSky*, a solution based on a quadtree data partitioning and skyline exploration. The overall approach is detailed in Fig. B.1. *QuadSky* consists of four main parts: spatial blocking (*QuadFlex*), pairwise comparisons, ranking the pairs (*SkyRank*), and labelling the pairs (the *SkyEx-** fam-

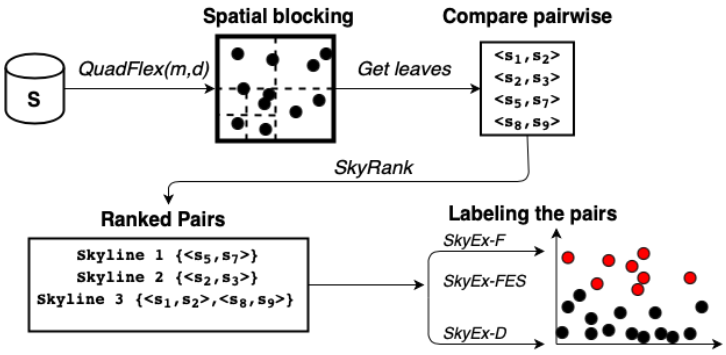


Fig. B.1: QuadSky approach

ily of algorithms). S contains all spatial entities. We propose *QuadFlex*, a quadtree-based solution that can perform the spatial blocking by respecting the distance between spatial entities and the density of the area. The output of *QuadFlex* is a list of leaves with spatial entities located nearby. Within the leaves, we perform the pairwise comparisons of the attributes. Then, we rank the compared pairs based on the skylines (concepts detailed in Sect. 6) using the *SkyRank* algorithm. In order to decide which pairs dictate a match and which not, we propose the *SkyEx-** family of algorithms (*SkyEx-F*, *SkyEx-FES*, and *SkyEx-D*) that finds which skyline level best separates the pairs that refer to the same physical spatial entity (the positives class) from the rest (the negative class). In the following sections, we detail each of the phases of *QuadSky*. We use the notations in Table B.1 (We will explain them gradually during the paper).

4 Spatial Blocking

Since spatial proximity is a strong indicator of finding a match, the first step is to group nearby spatial entities in blocks. Several generic blocking techniques have been discussed in [27, 28], but mostly based on textual attributes and not applicable to spatial blocking. We propose a quadtree-based solution (*QuadFlex*) that uses a tree data structure but also preserves the spatial proximity of spatial entities. A quadtree is a tree whose nodes are always recursively split into four children when the capacity is filled [29]. After the quadtree is constructed, the points that fall in the same leaf are nearby spatially. Hence, these leaves are good candidates to be spatial blocks. However, the existing quadtree algorithm needs to be adapted for spatial blocking.
First, a quadtree needs a capacity (number of points) as a parameter. The capacity is not a meaningful parameter for spatial blocking, while the density

Notation	Description
s	A spatial entity with a point p and a set of attributes $\{a_i\}$
S	A set of spatial entities $\{s_i\}$
Q	A *QuadFlex* structure used for spatial blocking
P	A set of pairs $\{\langle s_i, s_j \rangle\}$
δ_a	The similarity of a pair in terms of attribute a
$u(\langle s_i, s_j \rangle)$	The utility of a pair $\langle s_i, s_j \rangle$
$Skyline(k)$	A skyline of pairs $\{\langle s_i, s_j \rangle\}$ in the level k
K	The total number of skylines
k	A variable indicating the level of skyline
k_f	A k value fixed by *SkyEx-F* and *SkyEx-FES*
k_d	A k value fixed by *SkyEx-D*
P_k	Pairs of P associated with a skyline
P^+	A subset of pairs in P classified as positive
P^-	A subset of pairs in P classified as negative
$F1(k)$	The F-measure in the k^{th} level of skyline
μ_d	The mean of the distances between the two classes.
$\mu_d(k)$	The function measuring μ_d in in each k level of skylines
$\mu'_d(k)$	The first derivative of $\mu_d(k)$

Table B.1: Notations used throughout the paper

of the area is a better candidate. For example, if the area is too dense (e.g., city center), even though the capacity is not reached, a further split would be more beneficial. On the contrary, two points in the countryside (e.g., a farm) might be farther apart, but they still might be the same entity. Second, a quadtree does not limit the distance between points. Even though two points might be in an area that respects the density, if they are quite distant from each other, it is not necessary to compare them. The maximal distance between two points in a child is the diagonal of the area (all quadtree children are rectangular). We used m, the diagonal of an area, as a parameter that controls the distance of points rather than comparing all distances between all spatial entities. Finally, a quadtree splits into four children, and sometimes nearby points might fall into different leaves. We modify the procedure of the assignment of the points into a child by allowing more than one assignment. Fig. B.2 shows the modifications that we do to the construction of the traditional quadtree for our version *QuadFlex*. The traditional quadtree divides the area of each parent into four smaller areas, the children. A point belongs only to one child. In our modification, the area will split into 4 children in the same way as a quadtree (at 0.5 of the height and 0.5 of the width of the parent), but when we assign a point to a child, we will consider including

Algorithm B.1 QuadFlex algorithm

Input: A set of entities $S = \{s_i\}$, diagonal m, density d
Output: The leaves QuadFlex Q *Q.leaves()* ;

Create $Q(m, d)$ where Q has the dimensions of the bounding box of S
for each s in S **do**
    Q.insert(s) *// Insert s into the QuadFlex*
end for
return *Q.leaves()*

Method *insert (s)*
if *this.children* $\neq$ **then**
    Indexes $\leftarrow$ *getIndex(s)* *// Find where s belongs*
    for each i in *Indexes* **do**
        this.child[i].insert(s) *// Insert s to the children it belongs*
    end for
end if
if *this.diagonal* $> m$ **or** *this.density* $> d$ **then**
    Split the current object *this* into 4 children
end if
Indexes $\leftarrow$ *getIndex(s)*
for each i in *Indexes* **do**
    this.child[i].insert(s)
end for
return

Method *getIndex (s)*
Let *vertical-left* and *vertical-right* be the lines that pass at 0.25 and 0.75 of the width of *this*, respectively
Let *horizontal-up* and *horizontal-down* be the lines that pass at 0.25 and 0.75 of the height of *this*, respectively
if s is left of *vertical-right* and above *horizontal-down* **then**
    Indexes.add(1) *// s fits in child[1]*
end if
if s is right of *vertical-left* and above *horizontal-down* **then**
    Indexes.add(2) *// s fits in child[2]*
end if
if s is left of *vertical-right* and below *horizontal-up* **then**
    Indexes.add(3) *// s fits in child[3]*
end if
if s is right of *vertical-left* and below *horizontal-up* **then**
    Indexes.add(4) *// s fits in child[4]*
end if
return *Indexes*

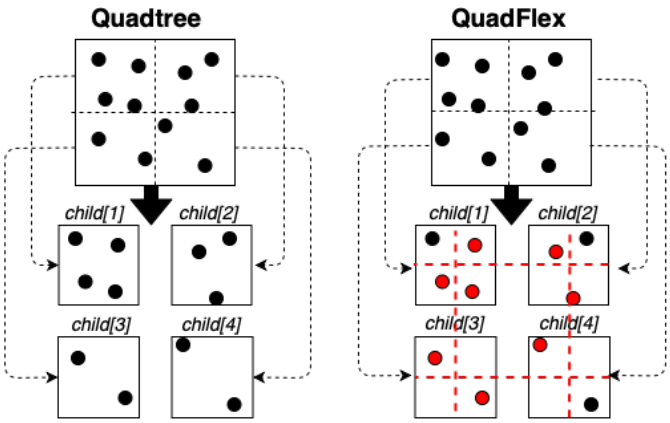


Fig. B.2: QuadFlex versus quadtree

points that fall shortly outside the border in the current child, too. For example, in Fig. 2, *QuadFlex* physically splits in the same way as the quadtree, but the red dashed line shows the area that will be considered for including neighboring points. The red points are in the overlapping regions and will be included in more than one child. Algorithm B.1 details the procedure for retrieving the spatial blocks with *QuadFlex*. The algorithm creates the root of the *QuadFlex* tree with the bounding box of the data and parameters m and d (line 1). Then, it inserts each spatial entity into the *QuadFlex* (line 3) and finally returns its leaves. The methods $insert(s)$ and $getIndex(s)$ are self calls on the *QuadFlex* object (*this*). The insertion procedure is similar to the traditional quadtree except that the constraint is not the capacity but the diagonal of the area m (maximal distance between points) and the density of the area d. Hence, if the diagonal of the *QuadFlex* is more than the distance m or the density is larger than our defined value d (line 12), the *QuadFlex*, similarly to a quadtree, will split into four children. However, in contrast to the traditional quadtree, a spatial entity might belong to more than one child. The method $getIndex(s)$ gets the list of indexes of the children where the new point will be assigned. *Even though Q splits into 4 children in the same way as a quadtree, the lines vertical-left, vertical-right, horizontal-up, and horizontal-down allow a logical overlap of the areas and thus, neighboring spatial entities will not be separated.*

5 Pairwise Comparisons

After the spatial blocking, we perform a pairwise comparison of spatial entities that fall in the same leaf. Next, we describe the metrics for different types of attributes.

Textual Similarity. We measure the textual similarity of spatial entities using the edit distance between the words. The Levenshtein distance [30] between string s_1 and string s_2 $d(s_1, s_2)$ is the number of edits (insertion, deletion, change of characters) needed to convert string s_1 to string s_2. We define the similarity as:

$$TextSim(s_1, s_2) = (1 - \frac{d(s_1, s_2)}{max(|s_1|, |s_2|)}) \tag{B.1}$$

Example B.1
Let us consider "Skippers Grill" and "Skippers Grillbar". The Levenshtein distance to convert "Skippers Grill" to "Skippers Grillbar" is 3 (3 insertions). The lengths of the first and the second string are 14 and 17 respectively. So, $TextSim$("Skippers Grill", "Skippers Grillbar") $= 1 - (3/max(14, 17) =$ 0.8235.

Note here that not all textual attributes can be handled similarly. String similarity metrics are usually appropriate for attributes like names, usernames, etc. Some other textual attributes require other metrics that need to be customized. In this paper, we consider the address as a specific textual attribute. The similarity between two addresses cannot be measured with Levenshtein, Jaccard, Cosine, etc. since a small change in the address might be a giant gap in the spatial distance between the entities. For example, "Jyllandsgade 15 9480 Løkken" and "Jyllandsgade 75 9480 Løkken" have a distance of 1 and Levenshtein similarity of 0.963, but they are 650 meters apart. However, "Jyllandsgade 15 9480 Løkken" and "Jyllandsgade 15 9480 Løkken Denmark" have a distance of 8 and Levenshtein similarity of 0.772, but they are the same building. In [11, 12] the address is considered as another textual attribute. In our case, we perform some data cleaning (removing commas, punctuation marks, lowercase, etc.), and then we search for equality or inclusion of the strings. We assign a similarity of 1.0 in the case of equality, 0.9 in the case of inclusion, and 0.0 otherwise.

Semantic Similarity. The similarity of fields like categories, keywords, or metadata cannot be compared only syntactically. Sometimes, several synonyms are used to express the same idea. Thus, we need to find a similarity than considers the synonyms as well. We use Wordnet [31] for detecting the type of relationship between two words and Wu& Palmer similarity measure

(*wup*) [32]. The semantic similarity between two spatial entities is the maximal similarity between their list of categories, keywords, or metadata. The semantic similarity of the spatial entities s_1 and s_2 is:

$$SemSim(s1, s2) = max\{wup(c_i, c_j)\} \tag{B.2}$$

where $c_i \in C_1$ and $c_j \in C_2$ and C_1 is the set of keywords of s_1 and C_2 is the set of keywords s_2.

Example B.2
Let us take an example of two spatial entities s_1 and s_2 and their corresponding semantic information expressed as keywords $C_1 = \{$"restaurant","italian"$\}$ and $C_1 = \{$"food","pizza"$\}$. The similarity between each pair is $wup($"restaurant","food"$) = 0.4$, $wup($"italian","food"$) = 0.4286$, $wup($"restaurant","pizza"$) = 0.3333$ and $wup($"italian","pizza"$) = 0.3529$. Finally, the semantic similarity of s_1 and s_2 is $SemSim(s1, s2) = max\{0.4, 0.4286, 0.3333, 0.3529\} = 0.4286$.

Date, Time, or Numeric Similarity. The similarity between two fields expressed as numbers, dates, times or intervals is a boolean decision (true or false). Even though the similarity of these fields relies only on an equality check, most of the effort is put in data preparation. For example, the different phone formats should be identified and cleaned from prefixes. Other data formats like intervals (opening hours) might require temporal queries for similarity, inclusion, and intersection of the intervals. In this paper, we do not compute the similarity between these attributes as we use them to construct the ground truth.

6 Ranking the Pairs

After the pairwise comparison, the pairs have n similarity values, one for each attribute. We denote as δ_a the similarity of two spatial entities for attribute a. For example, a pair $\langle s_1, s_2 \rangle$ is represented as $\{\delta_{a_1}, ..., \delta_{a_n}\}$. The problem that we need to solve is which $\langle s_i, s_j \rangle$ pairs indicate a strong similarity to be considered for a match. The related work solutions propose using a classifier [7, 8, 33] or experimenting with different thresholds [9, 18, 33]. We propose a more relaxed technique that uses Pareto optimality [34] for filtering the positive class. *A solution (x, y) is Pareto optimal when no other solution can increase x without decreasing y*. The points in the same Pareto frontier or skyline have the same utility. Widely used in economics and multi-objective problems, Pareto optimality is free of weights and similarity score functions.

In the context of entity resolution, the skylines provide a selection of points that are better than others, but without quantifying how much better. The pairs that refer to the same physical spatial entity (the positive class) are expected to have high values of δ, and consequently, form the first skylines. Under the assumption that the best values of δ belong to the pairs from the positive class, we label the pairs up to the k^{th} skyline as the positive class and the rest as the negative. *To the best of our knowledge, we are the first to propose a Pareto optimal solution for detecting matches for an entity linkage problem.*

Definition B.2. *An attribute a is positive discriminating if its similarity δ_a indicates a positive class rather than a negative.*

An example of a positive discriminating attribute is the similarity of name. A higher name similarity is more likely to indicate a match than a non-match. For example, the name similarity for *Mand & Bil* and *Mand og Bil* is 0.75, and for *Solid* and *Sirculus ApS* is 0.16 . Hence, the former pair has a higher probability of being a match than the second. Examples of negative discriminating attributes are the edit distance between two names. If the distance between the names is high, then the pairs are less likely to be a match.

Definition B.3. *The utility of a positive discriminating attribute a, denoted as u_a, is the contribution of the attribute similarity δ_a to reveal a match, using Pareto Optimality ($\delta_a \xmapsto{\text{Pareto Optimality}} u_a$).*

Each attribute similarity contributes to the labeling problem. Intuitively, a higher similarity δ_a of a has a higher utility than a lower value of δ_a. Hence, if $\delta_a(\langle s_1, s_2\rangle) > \delta_a(\langle s_3, s_4\rangle)$, then $u_a(\langle s_1, s_2\rangle) > u_a(\langle s_3, s_4\rangle)$.

Definition B.4. *The utility of a pair denoted as $u(\langle s_i, s_j\rangle)$ is sum of the utilities of each of the attributes. $u(\langle s_i, s_j\rangle) = \sum_{i=1}^{n} u_{a_i}$.*

Note that the utility of a pair is not the sum of the similarities of the attributes ($u(\langle s_i, s_j\rangle) \neq \sum_{i=1}^{n} \delta_{a_i}$) but the sum of their utilities ($u(\langle s_i, s_j\rangle) = \sum_{i=1}^{n} u_{a_i}$). Nevertheless, $u(\langle s_i, s_j\rangle) = \sum_{i=1}^{n} \delta_{a_i} = \sum_{i=1}^{n} u_{a_i}$ is a specific case.

Definition B.5. *A skyline of level k, Skyline(k), is the collection of pairs $\langle s_i, s_j\rangle$ of equal utility such that $u_{Skyline(k)} > u_{Skyline(k+1)}$.*

Obviously, *Skyline*(1) is the Pareto optimal frontier with the best values of δ_a. In order to continue with *Skyline*(2), the points of *Skyline*(1) are removed, and the frontier is calculated again. Every time we explore level k, the values in *Skyline*(k) are the ones with the highest utility. This means that *there is no other point in a lower level that can bring a higher utility to the positive class*. This procedure continues until all the pairs are ranked according to their skyline. Algorithm B.2 formalizes our proposed procedure Skyline Ranking

(*SkyRank*) for ranking the pairs. The input is the set of pairs P produced from the *QuadFlex* blocking technique and the number of skyline levels k that we will explore. We find the points with the best combinations of δ that dominate the rest of the points and, consequently, have a higher utility (line 3). Then, we put these points in P_k, which keeps the explored skylines and remove them from P (line 5). We stop when all the pairs are assigned to a skyline.

Algorithm B.2 Skyline Ranking (SkyRank)

Input: A set of pairs $P = \{\langle s_i, s_j \rangle\}$
Output: A set of pairs and their skyline $P_k = \{\langle s_i, s_j \rangle, k\}$;
1: $P_k \leftarrow \emptyset$
2: **while** $|P_k| < |P|$ **do**
3: Filter $Skyline(k) = \{\langle s_i, s_j \rangle\} \mid \forall \langle s', s'' \rangle \in P - \{\langle s_i, s_j \rangle\}$, $u(\langle s_i, s_j \rangle) > u\langle s', s'' \rangle\}$ *// Find the Skyline*
4: Add $Skyline(k)$ to P_k *// Move the skyline to* P_k
5: $P = P - Skyline(k)$
return P_k

After obtaining the ranking, we can assume that the pairs of the first few skylines are more likely to refer to the same physical entity than the rest.
6: **Assumption B.1.** *The probability that a pair is labeled positive is inversely proportional to its skyline level.*
The assumption considers that for all $\langle s_i, s_j \rangle$and$\langle s'_i, s'j \rangle$ in P such that $\langle s_i, s_j \rangle \in$ *Skyline(k)*, $\langle s'_i, s'_j \rangle \in$ *Skyline(k')* and $k < k'$, then $\langle s_i, s_j \rangle$ is more likely to be a match than $\langle s'_i, s'_j \rangle$.

7 Estimating k

In this section, we estimate the skyline level k that separates the positive from the negative class. We introduce two different methods for fixing the value of k: *threshold-based* (*SkyEx-F* and *SkyEx-FES*) and *unsupervised* (*SkyEx-D*).

7.1 SkyEx-F and SkyEx-FES

In contrast to the threshold-based methods used in entity resolution problems [12, 13, 18] where we have to find a threshold for each similarity of the attributes and then a threshold for the similarity function that aggregates the similarity scores, we have simplified our problem to only one parameter: k. We need to find the value of k that best separates the classes. As a measure of a "good model", we choose to use the *F-measure*, given that our data

tends to be unbalanced [35–37]. In the context of our problem, we define true positives *TP* as pairs that refer to the same physical entity and are correctly labeled as positives; true negatives *TN* as pairs referring to different physical entities and are correctly labeled as negatives; false positive *FP* as pairs that do not refer to the same physical entities but are wrongly labeled as positives; *FN* as pairs that refer to the same physical entity but are wrongly labeled as negatives. Thus, the precision is $p = \frac{TP}{TP+FP}$, the recall is $r = \frac{TP}{TP+FN}$ and *F-measure* $(F1) = 2\frac{p*r}{p+r}$.

Algorithm B.3 SkyEx-F

Input: A set of pairs $P = \{\langle s_i, s_j \rangle\}$
Output: A set of positive pairs P^+ and a set of negative pairs P^- ;
1: $P_k \leftarrow \emptyset$, $F \leftarrow \emptyset$
2: **while** $|P_k| < |P|$ **do**
Lines 3-5 as Algorithm B.2 ...
6: $P^+ \leftarrow P_k$
7: $P^- \leftarrow P$
8: Calculate $F1(k)$
9: Add $\langle k, F1(k) \rangle$ to F
10: Find k_f such that $F1(k_f) = max(F1(k))\ \forall k \in \{1, |F|\}$
12: $P^+ \leftarrow \bigcup_{k=1}^{k_f} Skyline(k)$
13: $P^- \leftarrow P - P^+$
return P^+, P^-

The higher the k, the more unlikely it is for a pair in the k^{th} skyline to belong to the positive class (Assumption 1). *SkyEx-F* explores the first skylines and stops at the value of $k = k_f$ that achieves the highest *F-measure*. To find k_f, we rank the pairs as in Algorithm B.2, but we add some extra calculations within the loop (lines 6-9) and find the optimal k_f (line 7) in Algorithm B.3. *SkyEx-F* calculates the *F-measure* for each skyline k by considering the pairs up to the k^{th} skyline as positive and the rest as negative. We add $F1(k)$ to the set F, which keeps track of the evolution of *F-measure* while exploring more skylines. We find k_f as the value of k that achieves the highest *F-measure* in F. The pairs from the first to the k_f level of skyline are labeled as positive and the rest as negative. Note that *SkyEx-F* explores all the skylines and then, finds the threshold k_f. However, we can optimize Algorithm B.3 by stopping at k_f before going through the full dataset P. Let us highlight some properties of p and r.

Property B.1. *The recall is a monotonically non-decreasing function with respect to the number of skylines k.*

Proof. The recall after k skylines is $r\ (k) = \frac{TP(k)}{TP(k)+FN(k)}$. While we move to the next, $k+1^{th}$ skyline, we label more pairs as positive, so the probability of finding true positives TP is higher. Thus, $TP(k+1) \geq TP(k)$. As for the denominator, it is always the same despite the skyline level because the true positives are fixed in P and are independent of our labelling. This means that if we find more true positives (TP), then we automatically decrease the false negatives (FN). Hence, $TP(k+1) + FN(k+1) = TP(k) + FN(k)$. We can then show that $\frac{TP(k+1)}{TP(k+1)+FN(k+1)} \geq \frac{TP(k)}{TP(k)+FN(k)}$ so $r\ (k+1) \geq r\ (k)$. □

Property B.2. *Given Assumption 1, the precision is a monotonically non-increasing function with respect to the number of skylines k.*

The precision is $\frac{TP}{TP+FP}$. However, $TP + FP$ is what our algorithm labels as positive, which means all the pairs belonging to skylines up to the k^{th} level. According to Assumption 1, FP increase at a higher rate than TP while moving to higher k values. A proof of monotonic decreasing precision for systems that rank the results considering their relevance (like our skylines) can be found in [38].

Theorem B.1. *The F-measure function with respect to the number of skylines k is increasing until a point or interval, and after that, it cannot increase again.*

Proof. Let us suppose that while moving deeper into the skylines, we found a peak point k or peak interval $[k_i, k_j]$ with $F1(k)$ as the corresponding *F-measure*. Note that for a peak interval the *F-measure* is constant. Since $F1(k)$ belongs to a peak point/interval, there exists a $F1(k+\epsilon)$ ϵ skylines after k such that $F1(k+\epsilon) < F1(k)$. Now, let us know suppose that we can find another optimum in $k+\delta$ such that $F1(k+\delta) > F1(k)$. Since $F1(k+\epsilon) < F1(k)$, consequently $F1(k+\delta) > F1(k) > F1(k+\epsilon)$. $F1 = 2\frac{p*r}{p+r}$ can be rewritten as $F1 = \frac{2}{\frac{1}{p}+\frac{1}{r}}$. So, we can rewrite: $\frac{2}{\frac{1}{p(k+\delta)}+\frac{1}{r(k+\delta)}} > \frac{2}{\frac{1}{p(k)}+\frac{1}{r(k)}} > \frac{2}{\frac{1}{p(k+\epsilon)}+\frac{1}{r(k+\epsilon)}}$. Using Property 2, $p(k+\delta) \leq p(k+\epsilon)$, so: $\frac{2}{\frac{1}{p(k+\epsilon)}+\frac{1}{r(k+\epsilon)}} \geq \frac{2}{\frac{1}{p(k+\delta)}+\frac{1}{r(k+\epsilon)}}$, which means that: $\frac{2}{\frac{1}{p(k+\delta)}+\frac{1}{r(k+\delta)}} > \frac{2}{\frac{1}{p(k+\delta)}+\frac{1}{r(k+\epsilon)}}$. According to Property 1, this inequality cannot hold, because $r(k+\delta) \geq r(k+\epsilon)$. Thus, our assumption of $F1(k+\delta) > F1(k)$ cannot hold and $F1(k)$ remains the highest value of *F-measure*. □

Theorem 1 ensures that once we find the peak in the *F-measure* function, we can stop finding all the skylines and label the pairs accordingly. Consequently, we can allow Algorithm B.3 to stop early. The modifications of Algorithm B.3 are reflected in Algorithm B.3a. We use the same procedure as in Algorithm B.3, but we do not need to keep track of each of the skylines and their corresponding *F-measures*. Rather, we only keep the previous *F-measures*

Algorithm B.3a SkyEx-F Early Stop (SkyEx-FES)

Input: A set of pairs $P = \{\langle s_i, s_j \rangle\}$
Output: A set of positive pairs P^+ and a set of negative pairs P^- ;

1: $P_k \leftarrow \emptyset$, $F_{previous} \leftarrow 0$
2: **while** $|P_k| < |P|$ **do**
    Lines 3-8 as Algorithm B.3...
9: **if** $F1(k) < F_{previous}$ **then**
10: **break**
11: **else**
12: $F_{previous} \leftarrow F1(k)$
13: **end if**

15: $P^+ \leftarrow \bigcup_{k=1}^{k_f} Skyline(k)$
16: $P^- \leftarrow P - P^+$
    return P^+, P^-

in $F_{previous}$. While moving to the next skyline, we calculate the *F-measure* and the first time we notice a drop (line 9), we stop the loop (line 10) and return both classes separated by the current k (lines 7-8)). Otherwise, we update $F_{previous}$ to the current *F-measure* (line 12) and continue the search for the optimal k.

7.2 SkyEx-D

The methods described in the previous sections assume that the labels of the pairs are present. In this section, we assume no information about the labels, and thus, we propose a heuristic for fixing the value of k. The heuristic is based on the distance between the positive and the negative class. We refer to the k discovered by *SkyEx-D* as *distance-based k* or k_d. Our classes are not characterized by a small intra-class distance. Various patterns can reveal a positive class; for example, a similar name but different category or similar category and similar address, etc. Thus, the positive pairs, positioned in the first skylines, are scattered and do not necessarily form a cluster. However, they can still be separated from the rest, considering the distance to the negative class. Theoretically, the inter-class distance stays small when we are in the first skylines (potential positives), then starts to increase while we move into later skylines and finally falls again when we enter the deeper skylines (potential negatives). *SkyEx-D* notices the increase of the inter-class distance and sets k_d accordingly. In order to have an approximation of the inter-class

distance, we use the mean and denote it as μ_d as in Eq. 3.

$$\mu_d = \frac{\sum d(p^k, p^{-k})}{|P_k|} \tag{B.3}$$

where $|P_k|$ is the number of pairs from the 1^{st} to the k^{th} skyline, p^k is a pair in P^k, p^{-k} is a pair in $P - P^k$, and $d(p^k, p^{-k})$ is the distance between p^k and p^{-k}.

In order to fix k_d, we monitor the value of μ_d while moving deeper into the skylines. We denote by $\mu_d(k)$ the function of μ_d with regard to k. We use the first derivative of $\mu_d(k)$, denoted as $\mu'_d(k)$, to find the points where the $\mu_d(k)$ function decreases. The intuition behind this approach is that in the beginning, the distance $\mu_d(k)$ starts increasing, which means that the first derivative has a positive slope ($\mu'_d(k) > 0$). Later, we enter the "grey area", where there is a mix of potential positives and potential negatives. This is where we need to stop because we might lose precision if we continue further. In order to find the "grey area", we note when the first derivative changes its slope to negative. In order to calculate $\mu'_d(k)$, we estimate the value of $\mu'_d(k)$ in each point k as in Eq. 4:

$$\mu'_d(k) = \frac{\partial}{\partial k} \approx \frac{\mu_d(k+1) - \mu_d(k)}{1} \tag{B.4}$$

In order to not be sensitive to small fluctuations in $\mu'_d(k)$, we smoothen slightly $\mu'_d(k)$ with Gaussian function ($\frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2 / 2\sigma^2}$) using a small window. Then we monitor when $\mu'_d(k)$ decreases for the first time and we set k_d accordingly. We modify Algorithm B.2 to accommodate this approach. We calculate $\mu'_d(k)$ for each point of k in line 7. Then, we have to find the first negative value of the smoothened $\mu'_d(k)$ (line 9) and fix k_d accordingly (line 10). Finally, we return the classes defined by k_d in lines 16-17.

Summary. Algorithm B.4 estimates the skyline level k that best separates the positive class from the negative class. *Similarly to clustering techniques that use heuristics to estimate their parameters, SkyEx-D uses the distance of the positive class from the rest as an indicator of class separability*. However, in contrast to clustering metrics, which focus on the robustness of clusters, this is not a requirement for the *SkyEx-** family of algorithms. *The positive pairs do not show similar patterns, but rather similar utilities, which can be better captured by skylines* (see Sect. 9.9). Experimentally, we show that our inter-class distance approach estimates k_d very close to k_f without loosing in *F-measure*. In contrast to techniques that use a scoring function, the *SkyEx-** family of algorithms abstracts the concept of utility. Thus, no weights or similarity function is needed. Even though the positive class can be characterized by various patterns of attribute similarities, the *SkyEx-** family of algorithms can

Algorithm B.4 SkyEx-D

Input: A set of pairs $P = \{\langle s_i, s_j \rangle\}$
Output: A set of positive pairs P^+ and a set of negative pairs P^- ;

```
1: P_k ← ∅
   Lines 2-6 as Algorithm B.2...
7: Calculate μ'_d(k) in each k
8: while k < k_last do
9:    if smooth(μ'_d(k)) < 0 then
10:      k_d ← k
11:      break
12:   else
13:      k ← k + 1
14:   end if

15: P+ ← ⋃_{k=1}^{k_d} Skyline(k)
17: P− ← P_k − P+
    return P+, P−
```

still group together the positive class based on the high utility, while a clustering technique would instead focus in grouping each pattern separately, without putting the positive-class pairs together into one cluster. Moreover, the flexibility of the *SkyEx-** family of algorithms makes it applicable to all problems where the expert knowledge on the contribution of the attributes is missing. Finally, the *SkyEx-** family of algorithms does not learn any behavior, so there is no risk of overfitting.

8 Complexity Analysis of QuadSky

In this section, we discuss the time complexity of our algorithms and of our *QuadSky* solution.

QuadFlex deals with points (not regions); thus, it behaves similarly to a point quadtree. *QuadFlex* splits the same way as a quadtree, but in contrast to the quadtree, the points can be assigned to more than one child. We construct the *QuadFlex* structure only for forming the blocks. Hence, the construction complexity is of interest to us. Let us denote by $|S|$ the number of points in S, c the smallest distance between any two points, and D_1 and D_2 the dimensions of the initial area containing all the points. Let us first estimate the depth of *QuadFlex*. The distance c of any two points p_1 and p_2 in *QuadFlex* is always less than the diagonal of the node they belong in. Given that *QuadFlex* allows neighboring points to be included in more than one child, this

calculation needs to be modified. The physical diagonal of the initial (level 0) node is $\sqrt{D_1^2 + D_2^2}$. The diagonal of level i is $\frac{\sqrt{D_1^2+D_2^2}}{4^i}$. To modify the calculation, we estimate the logical diagonals if *QuadFlex* would physically expand to accommodate neighboring points, so: $c \leq \frac{\sqrt{\frac{3D_1^2}{2}+\frac{3D_2^2}{2}}}{4^i}$ Now, isolating i out of this equation results in $i \leq \log_4 \frac{\sqrt{\frac{3}{2}(D_1^2+D_2^2)}}{c} = \log_4 \sqrt{\frac{3}{2}} + \log_4 \frac{\sqrt{D_1^2+D_2^2}}{c}$. $\log_4 \sqrt{\frac{3}{2}} \approx 0.14$ so we can discard it (less than one level): $i \leq \log_4 \frac{\sqrt{D_1^2+D_2^2}}{c}$. For estimating the maximal depth, we need to add one more level (root) so the depth is estimated as $\log_4 \frac{\sqrt{D_1^2+D_2^2}}{c} + 1$. Finally, for constructing *QuadFlex*, the complexity is $O(|S|(\log_4 \frac{\sqrt{D_1^2+D_2^2}}{c} + 1))$.

SkyRank requires calculating the Pareto frontiers, which is time-consuming. In the typical case, comparing the pairs P resulting from *QuadFlex* in terms of all d dimensions has a $O(2^{|P|^d})$ time complexity [39], which is not scalable. *SkyRank* uses the method proposed in [40], which first scales down the d-dimensional domain and then pre-filters the data using a lattice. This yields a time complexity of $O(|P|^2)$ for the first skyline. For the total number of K skylines, the complexity is $O(K|P|^2)$.

SkyEx-F calculates the metrics while adding the next skyline to the positive class; thus, these calculations do not add any complexity. Finally, we perform a linear search on F to find the skyline with the highest F-measure. The size of F is equal to K, so the complexity is $O(K|P|^2 + K)$.

SkyEx-FES stops earlier than *SkyEx-F*, avoiding a big part of the time-consuming Pareto calculations. Given that the best pairs usually are focused on the first skylines, the cut-off $k \ll K$. Moreover, according to Theorem 1, we do not need to store F, so we avoid the linear search for the best F-measure. The complexity is $O(k|P|^2)$.

SkyEx-D uses all K Pareto calculations and then, in order to estimate the cut-off k_d, it computes the distance between the positive class and the rest. *SkyEx-D* creates a matrix where the rows are the positive class P^+ and the columns are the negative class data points $|P| - P^+$, so the complexity is $P^+ * (|P| - P^+)$. $P^+ * (|P| - P^+) = P^+ * |P| - (P^+)^2$ is the equation of a vertical parabola that opens downwards $-ax^2 + bx + c$, with the maximal value at the vertex $(-\frac{b}{2a})$. In our case, the maximum of $P^+ * (|P| - P^+)$ is at $\frac{|P|}{2}$, resulting in a maximal complexity of $\frac{|P|^2}{4}$. For each skyline k in K, the maximal complexity is $\frac{|P|^2}{4}$, thus, $K\frac{|P|^2}{4}$ for all. Note here that $K \ll |P|$ so it is far from a cubic complexity. SkyEx-D computes the mean distance μ_d for each k, which can already be done within the $\frac{|P|^2}{4}$ complexity. Then, we compute the derivative μ'_d on the means, which has a linear complexity

in K. Finally, we need another partial scan until k_d ($k_d \ll K$) where the derivative μ'_d becomes negative for the first time. Hence, the total complexity is $O(K|P|^2 + K\frac{|P|^2}{4} + K + k_d) = O(\frac{5K}{4}|P| + K + k_d)$.

Summary . *QuadFlex* has $O(n \log n)$ complexity, the pairwise comparisons have a linear $O(n)$ complexity, while the *SkyEx-** family of algorithms have a quadratic complexity $O(n^2)$. However, there is a theoretical risk of a cubic complexity in *SkyEx-F* and *SkyEx-D* if the number of skylines $K = |P|$. This means that each skyline in K contains only one pair of entities, which theoretically can happen but almost never happens in practice. Thus, the algorithms have quadratic complexity in the average case. *SkyEx-D* has the highest complexity, followed by *SkyEx-F* and *SkyEx-FES*. Overall, *QuadSky* has a quadratic complexity.

9 Experiments

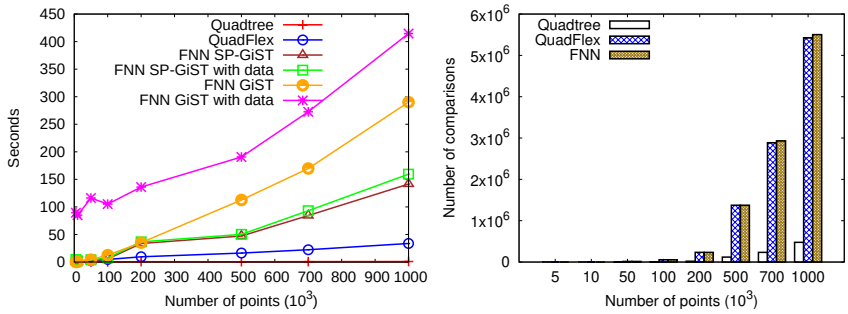
9.1 Dataset Description

The spatial entities that will be used in these experiments originate from four sources, namely Google Places (GP), Foursquare (FSQ), Yelp, and Krak. Krak (www.krak.dk) is a location-based source that offers information about companies, enterprises, etc. in Denmark and is also part of Eniro Danmark A / S., which publishes The Yellow Pages. The data is obtained by using the available APIs and the algorithm detailed in [41]. The dataset consists of 75,541 spatial entities where 51.50% comes from GP, 46.22% from Krak, 0.03% from FSQ, and 2.23% from Yelp (see Supp. Material Annex A for the spread of these spatial entities on the map). The dataset is 69 MB. For a 100 m blocking, there are 35,521 spatial entities that have at least one positive match in the dataset, resulting in 27,102 pairs that need to be discovered. 7,795 of these pairs are within the same source, which shows that none of these sources are free of duplicates. 3,546 of the same-source links come from GP, 3,789 from Krak, and 460 from Yelp. As for the different-source links, all the sources overlap with each other, but the highest overlap of 17,405 pairs (90% of different-source links) comes from Krak and GP.

9.2 QuadFlex Performance

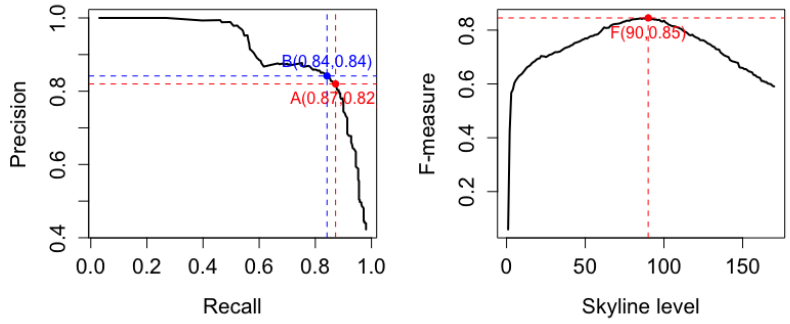
In this section, we compare the performance of *QuadFlex* to the quadtree, Fixed Radius Nearest Neighbors algorithm [42] (FNN), and having no index at all (No-Index). FNN finds the neighbors that fall within a fixed radius from each point. *QuadFlex* and the quadtree algorithm are implemented in Java, while FNN is run on a Postgres database (https://www.postgresql.org)

using spatial indexes: GiST (optimized C implementation of B-trees and R-trees) and SP-GiST
(optimized C implementation of quadtrees and k-d trees). Our dataset contains 75,541 entities in the North Denmark region (around 16 towns, 7,933 km^2), so the average density is not high, even though there are areas with high density. A high data density means more pairs to compare. To test our *QuadFlex* on different data densities, we simulate up to 1,000,000 random points from Aalborg (139 km^2). Fig. B.3 shows the comparison of quadtree, *QuadFlex* and FNN in terms of execution time (Fig. B.3a) and number of comparisons (Fig. B.3b). The FNN versions with data are computed on the database, and then the pairs are loaded back to the java implementation. The quadtree has the lowest execution time, followed by *QuadFlex*. FNN SP-GiST is comparable and sometimes even better than *QuadFlex* for small datasets. However, when the size of the dataset increases, *QuadFlex* maintains an execution time that is eight times less than FNN GiST and 3 times less than FNN SP-GiST. FNN with SP-GiST index outperforms FNN GiST for all dataset sizes. No-Index was very inefficient, up to 848 times slower than FNN Gist with data, and up to 368,095 times slower than *QuadFlex*. Given that No-Index would have dwarfed the other curves, it is not part of Fig. B.3a, but instead, refer to Fig. 2 in Supp. Material, Annex B. As for the number of comparisons, *QuadFlex* enumerates 12 times more comparisons than quadtree. Moreover, *QuadFlex* contains almost all (99.99%) comparisons of FNN, compared to the quadtree that contains only 10% of FNN. Furthermore, given that the scalability of *QuadFlex* is better than FNN, and *QuadFlex* is independent of the database implementations, the loss of around 0.01% of comparisons is insignificant.



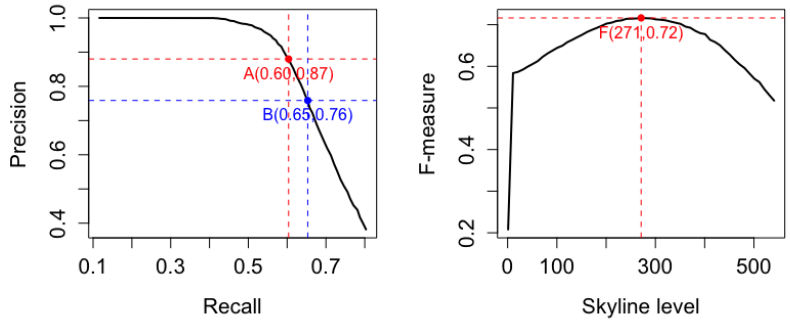
(a) Execution time **(b)** Number of comparisons

Fig. B.3: Comparing quadtree, QuadFlex and FNN



(a) precision and recall **(b)** F-measure

Fig. B.4: SkyEx-F performance on D_{sample}

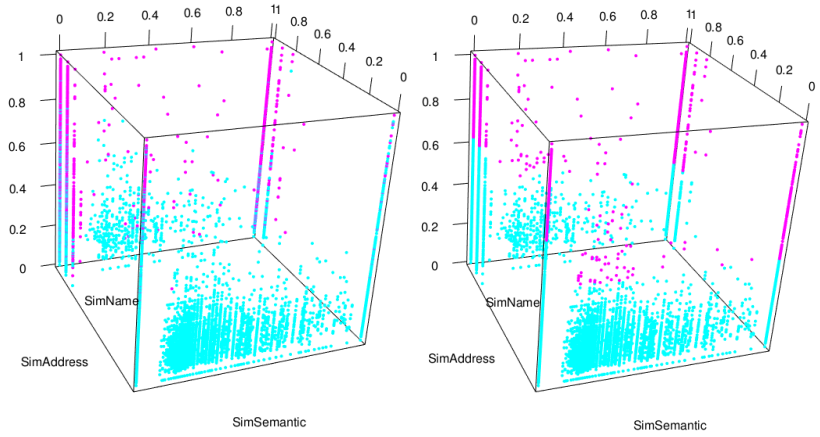


(a) precision and recall **(b)** F-measure

Fig. B.5: SkyEx-F performance on D_{full}

9.3 SkyEx-F Results

We ran *QuadFlex* with 100 m and no density restriction, and we obtained 777,452 pairs (1426 MB). Having the same website or phone is a strong indicator of a match, so we use these attributes to infer the label. We refer to this labeling as *automatic labeling*. However, cases with different phone number or website but still the same entity, or same phone number but different entity might occur. Hence, we manually checked the labels of a sample of 1,500 pairs of entities (1552 kB). We will refer to the sample of manually checked pairs as D_{sample} and to the full dataset as D_{full}. Checking the labels manually on the full dataset of 777,452 pairs is unfeasible. Hence, we checked around 10,000 of the pairs, and for the rest, we rely on automatic labeling.
The results of *SkyEx-F* on D_{sample} and D_{full} are presented in Figs. B.4 and B.5.



(a) Actual classes **(b)** SkyEx classes

Fig. B.6: Positive (in pink) versus negative (in sky blue) classes for actual (a) and SkyEx-F (b) results

The curves in Figs. B.4a and B.5a shows the evolution of p (y-axis) and r (x-axis) while we move from one skyline to the next. *The more we explore, the more likely it is to retrieve more true positives and thus, improve the r*. However, *the more we explore and label pairs as positives, the more likely it is to increase the number of false positives, so the p degrades*. The algorithm explores several trade-offs; for example the points A and B are among the best. The point A with 0.87 p and 0.82 r in Fig. B.4a is the same best point in terms of *F-measure* as well, so that is where *SkyEx-F* will fix k_f. Fig. B.4b shows the levels of the skyline, and the value of *F-measure* achieved. The highest value is 0.85 that corresponds to $k = 90$. The evaluation on the full dataset yields lower values (*F-measure* of 0.72) compared to the sample (*F-measure* of 0.85), which might be a simple consequence of automatic labeling. Point A has 0.6 r and 0.87 p, while B offers a higher r of 0.65 but a lower p of 0.76 (Fig. B.5b). To have an idea of the real classes in D_{full} and the skylines, we plotted their distribution in Fig. B.6 (the actual positive classes in pink and the negative ones in sky blue). It is noticeable that the positive class pairs are allocated in the highest values of the dimensions. Despite the differences between both plots, *SkyEx-F* shows promising results in separating the positive class from the negative one with 0.6 r and 0.87 p.

9.4 Experimenting with Different QuadFlex Parameters

So far, we used *QuadFlex* blocking technique with 100 meters and no density restriction. In this section, we will evaluate our approach *QuadSky* for different blocking parameters.

Meters	1	20	40	60	80	100
Total	41053	118437	226331	372553	557421	777452
% of TP	17.11%	19.88%	11.28%	7.06%	4.82%	3.49%
Prec.	0.67	0.80	0.85	0.85	0.88	0.87
Rec.	0.60	0.69	0.65	0.64	0.61	0.61
F1	0.64	0.75	0.74	0.73	0.72	0.72

Table B.2: *SkyEx-F* results for different m

Changing m, no density limit In this experiment, we test different values of m used in *QuadFlex* for creating spatial blocks. We test m values of 1, 20, 40, 60, 80, and 100 meters. The size of the dataset for each of them is presented in Table B.2. The spatially closeby points are likely to be a match. Hence, the percentage of the true positives is generally higher for smaller values of m. An interesting case is $m = 1$, where the percentage of the true positives (TP) is lower than $m = 20$. One would expect that points that are 1 meter apart would unquestionably be a match. However, this is not always the case. Shopping malls, buildings that host several companies, etc. are characterized by the same coordinates but not necessarily the same spatial entities. The results for different values of m are presented in Table B.2 (see the precision-recall graphs for all cut-offs in Supp. Material, Annex C). For all cases, the r is higher than 0.6. The p is higher than 0.8 for all values of m, except $m = 1$, where the p is 0.67. For $m = 1$, the positive and negative class are mixed, thus *SkyEx* loses a bit in p. This is also an argument against the works that merge arbitrarily points that are 5 m apart. *Spatial proximity is not a definitive indicator of a match.*

Changing d, m $\leq$ 100. We experiment with different values of density d and its effect on the results. The size of the dataset, the percentage of the true positives, and the results in terms of precision, recall, and *F-measure* are in Table B.3 (see the precision-recall graphs for all cut-offs in Fig. 9 in [14]). When the density is smaller, we force *QuadFlex* to split further and create smaller blocks. Thus, the number of pairs reduces. Note that, on the contrary, the percentage of the true positives (TP) increases. Indeed, further splits allow us to create better blocks containing a higher percentage of TP. However, when the density limit increases above $\frac{30s}{1000m^2}$, fewer and fewer blocks are split further, so the dataset size and the percentage of the TP do not vary significantly. In all the cases, the r stays above 0.61 and the p above 0.87. A slightly better p (0.88) and r (0.63) is achieved in the case of a density of $\frac{10s}{1000m^2}$ (the lowest parameter). *SkyEx-F adapts very well in finding the correct classes even when the size of blocks changes and even when the percentage of the true positives over the true negatives varies.*

Den.	$\frac{10s}{1000m^2}$	$\frac{20s}{1000m^2}$	$\frac{30s}{1000m^2}$	$\frac{40s}{1000m^2}$	$\frac{50s}{1000m^2}$	$\frac{60s}{1000m^2}$
Total	290653	590583	711423	754195	770987	776664
% of TP	8.61%	4.57%	3.81%	3.59%	3.51%	3.49%
Prec.	0.88	0.88	0.87	0.87	0.87	0.87
Rec.	0.63	0.61	0.61	0.61	0.61	0.61
F1	0.74	0.74	0.72	0.72	0.72	0.72

Table B.3: *SkyEx-F* results for different d

9.5 SkyEx-FES Optimization

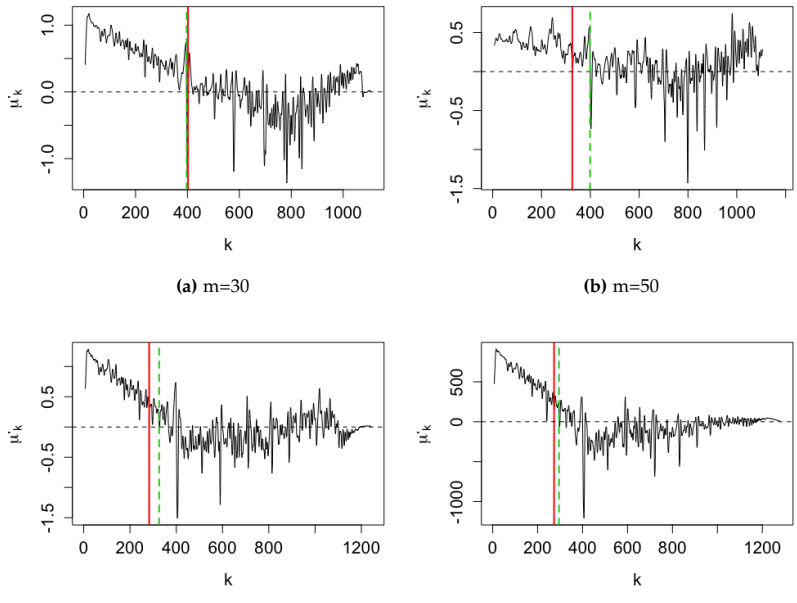
Given the theoretical guarantee in Theorem 1, we can stop *SkyEx-F* earlier as described in Algorithm B.3a. We ran *SkyEx-FES* for spatial entities that are 30, 50, 80, and 100 m apart. For all the cases, *SkyEx-FES found the same k_f values as SkyEx-F exploring only 27% of the skylines on average.* The comparison regarding the number of iterations is shown in Table B.4. For spatial entities that are 30 m, 50 m, 80 m, and 100 m apart, *SkyEx-FES* finds the optimal k_f exploring 36%, 27%, 23%, and 22% of the skylines, respectively. Moreover, our theoretical guarantee that the *F-measure* function has only one optimum can also be noticed in Figs. B.4b and B.5b.

Distance	**30 m**	**50 m**	**80 m**	**100 m**
Number of pairs	168193	293833	557421	777452
% of TP	14.76%	8.8%	4.82%	3.49%
***SkyEx-F* skylines**	1113	1182	1228	1228
***SkyEx-FES* skylines**	403	327	284	274

Table B.4: Skyline explorations of *SkyEx-FES* compared to *SkyEx-F* for pairs that are 30, 50, 80, and 100 m apart

9.6 SkyEx-D Performance

In these experiments, we use *SkyEx-D* (Algorithm B.4) to set k_d and evaluate our results in terms of *F-measure*. We apply *SkyEx-D* on spatial entities that are 30, 50, 80, and 100 meters apart (see the dataset details in Table B.4). We calculate the first derivative (μ'_d) in each point as in Algorithm B.4. The smoothed $\mu'_d(k)$ with respect to k are presented in Fig. B.7. The red solid line shows the value of k_f, while the green dashed line represents k_d found by *SkyEx-D*. We note when $\mu'_d(k)$ is negative for the first time and set k_d accordingly. In the case of spatial entities that are 30 m apart (Fig. B.7a), k_d is only 5 skylines apart from k_f but 73 skylines for 50 m. These values of k_d are discovered using the first derivative (Eq. 4, Sect. 7.2). We illustrate the trend of μ_k while increasing k, which means that we explore deeper skylines and

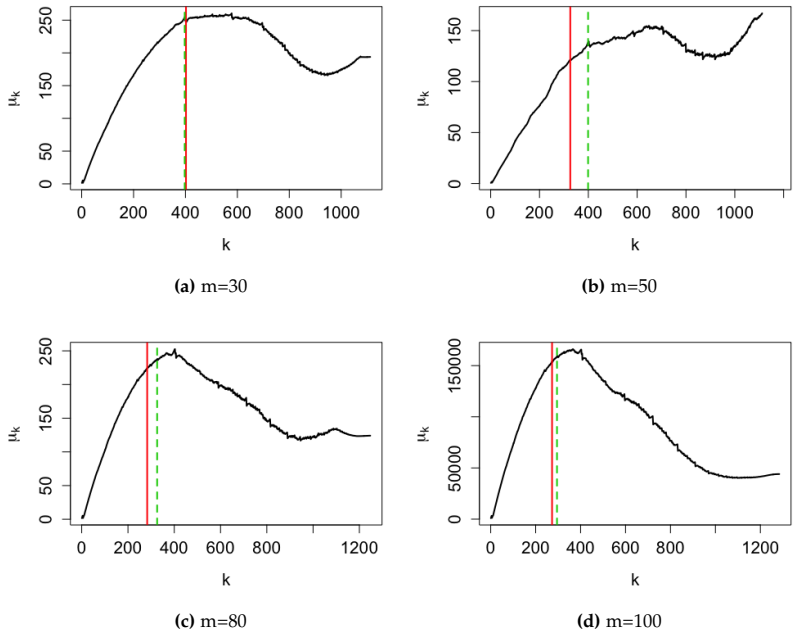


(a) m=30 **(b)** m=50

(c) m=80 **(d)** m=100

Fig. B.7: Setting k_d using μ'_d

examine more pairs that are less likely to be a match. The distance from the positive class to the negative is smaller in the beginning because the mean μ_k is biased by the close points. While we increase k, μ_k increases, meaning that the classes are becoming more and more distinguishable from one another. The high values of μ_k show a high distance between the classes. For spatial entities that are 80 m and 100 m apart, μ_k starts dropping faster than for those that are 30 m and 50 m apart (Fig. B.8). This observation can be justified by the fact that closeby entities are more difficult to classify, so the "grey" area of the potential cut-off is larger. However, *SkyEx-D* detects the first decrease in μ_k from the first derivative and fixes k_d. Graphically, this point coincides with the beginning of the "grey" area. Even though k_d is sometimes fixed far from k_f (m=50), the corresponding *F-measures* are almost the same (Fig. B.9). The red line in Fig. B.9 corresponds to k_f and the green line to k_d. The difference in *F-measure* is 0.002 for 30 m, 0.009 for 50 m, 0.002 for 80 m, and 0.004 for 100 m. Thus, *the difference in F-measure for the classifying the pairs using k_d instead of k_f is always less than 0.01.* This means that our *SkyEx-D*, even though fully unsupervised, is almost optimal in terms of *F-measure*.
In terms of precision, recall and *F-measure*, *QuadSky* with *SkyEx* in [14], *QuadSky* with *SkyEx-F*, and *QuadSky* with *SkyEx-FES* report the same values. How-



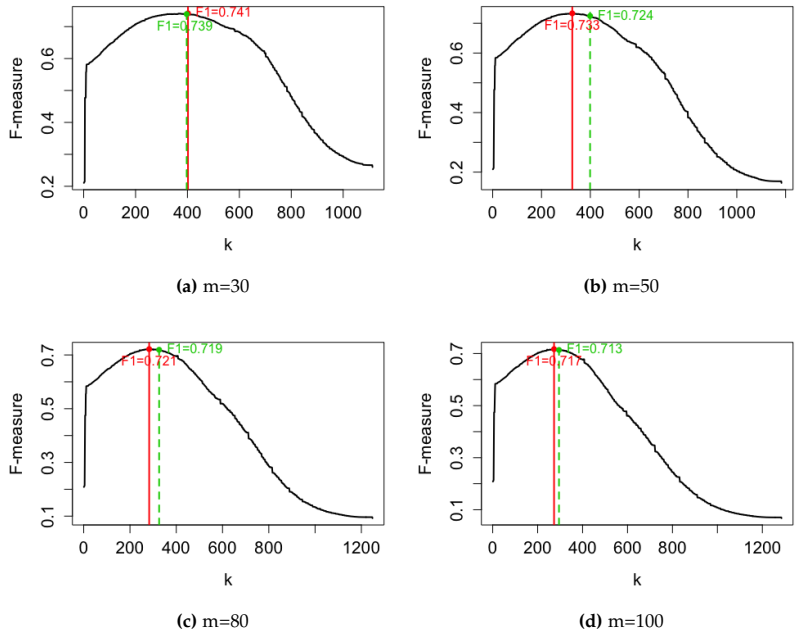
(a) m=30 **(b)** m=50 **(c)** m=80 **(d)** m=100

Fig. B.8: $\mu_d(k)$ function with respect to k

ever, the underlying algorithms are different. *SkyEx* in [14] needs the threshold k to separate the skylines, whereas for *SkyEx-F* and *SkyEx-FES*, there is no need for specifying k because the algorithms will fix it through the skyline explorations (only 30% of the skylines for *SkyEx-FES*). *QuadSky* with *SkyEx-D*, being fully unsupervised, might yield different results. The optimal scenario is if it fixes k_d as the k_f.

9.7 Comparison with Baselines

Even though there are several papers in spatial data integration, the works of [11–13] are the most similar to ours, as the rest of the related work considers only spatial objects, not spatial entities, or uses supervised learning techniques. We will compare *QuadSky* to Berjawi et al. [12], Morana et al. [13], and Karam et al. [11]. Berjawi et al. [12] propose Euclidean distance for the geographic coordinates and Levenshtein similarity for all other attributes. The similarities added together to a global similarity. The attributes mentioned in the paper are the name and the phone. However, since the phone is part of our automatic labeling, it can not be used in the algorithm as well. The authors consider pairs with *score* $\geq$ 0.75 as a match with high confidence. We use this threshold but also try other thresholds that might yield better



(a) m=30 **(b)** m=50 **(c)** m=80 **(d)** m=100

Fig. B.9: *F-measure* values for different k

results (the versions with the suffix *-Flex*). We compare against two versions proposed by the authors: name + address + geographic coordinates (V1) and name + geographic coordinates (V2). Morana et al. [13] suggest filtering entities that share the same category or a token in the name. Then these entities are compared using the Euclidean distance for the coordinates, Levenshtein for the address and name, and Resnik similarity (Wordnet) for the category. Attributes like address, phone, etc. are considered secondary, so they are given $\frac{1}{3}$ of the weight in the similarity score function, while name, category, and geographic proximity carry $\frac{2}{3}$ of the weight. The authors show top k matches for each entity to the user to decide. Karam et al. [11] starts with filtering spatial entities that are 5 m apart. Then, the similarity of the name is measured with Levenshtein distance, the geographic similarity with Euclidean distance and the keywords are compared semantically. In order to decide which pairs to match and which not, the similarities are fused using belief theory [26].

The results using D_{full} and D_{sample} are presented in Table B.5. In general, all the methods performed better in D_{sample} due to the better quality of the labels. Berjawi et al.(V2) [12] yields reasonable results, the second best after *QuadSky*, with an *F-measure* of 0.63 in D_{full} and 0.74 in D_{sample}. If we allow

flexible thresholds, Berjawi et al.(V2) [12]-*Flex* in D_{full} finds the same best threshold of 0.75, whereas in D_{sample} the threshold of 0.65 yields better results, increasing the *F-measure* from 0.74 to 0.79 (see Supp. Material, Annex D for all the thresholds and their results). To compare with Morana et al. [13], we tried all values k from 1 to the maximal matches for a single point (see Fig. 10 in [14]). The highest value of *F-measure* corresponded to a p of 0.39 and a r of 0.60. The behavior of Morana et al. [13] in D_{sample} is similar; the best value of *F-measure* was achieved for $k = 3$ and results are similar to those in D_{full}. The work of Karam et al. [11] achieves the highest r of 0.73 but a very low value of p of 0.23 for D_{full}. As a result, the *F-measure* is only 0.47. However, in D_{sample}, the method performs better overall (*F-measure* =0.6).
The QuadSky versions provide the best trade-off between p and r, and thus, the highest F-measure in both datasets. In D_{sample}, *QuadSky* with *SkyEx-F* and *QuadSky* with *SkyEx-D* achieve the best r compared to all baselines. What is more important, *QuadSky with SkyEx-D, even using an unsupervised algorithm, is still better than the threshold-based baselines.* The highest p values for both datasets is achieved by Berjawi et al.(V1) [12] but a very low r and poor model performance overall. In fact, models that achieve extreme values (high precision-low recall or low precision-high recall) are not a viable solution because they are either too restrictive or too flexible, and their predictability is poor. Berjawi et al. [12](V2)-*Flex* assumes the same weights for all similarities, and the reported values of p and r are good. However, *the behaviors of the pairs can be of all types. QuadSky can capture these different behaviors better than a simple sum would*.
Regarding the complexity of the baselines, we cannot judge in terms of the blocking techniques because there are no details on whether the authors used an index to create the blocks. However, as we show in Fig. B.3, the available FNN solutions in Postgres still do not scale as well as our *QuadFlex*. Therefore, we perform better in the blocking step. The pairwise comparison has a linear complexity for all baselines and our solution. As for the labeling,

	D_{full}			D_{sample}		
Approach	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
Berjawi et al.(V1) [12]	**0.93**	0.26	0.41	**1.00**	0.27	0.43
Berjawi et al.(V1) [12]-*Flex*	0.87	0.50	0.63	0.79	0.42	0.55
Berjawi et al.(V2) [12]	0.73	0.56	0.63	0.97	0.60	0.74
Berjawi et al.(V2) [12]-*Flex*	0.73	0.56	0.63	0.82	0.76	0.79
Morana et al. [13]	0.39	0.60	0.47	0.33	0.60	0.43
Karam et al. [11]	0.23	**0.73**	0.35	0.54	0.68	0.60
QuadSky with *SkyEx-F*	0.87	0.60	**0.72**	0.87	**0.82**	**0.85**
QuadSky with *SkyEx-D*	0.85	0.62	0.71	0.87	**0.82**	**0.85**

Table B.5: Comparison with the baselines

the baselines do not need the quadratic complexity induced by our skylines. Our *SkyEx-** family of algorithms run for 1 minute in D_{sample} and up to 2 hours in D_{full} with 777,452 pairs. Nevertheless, the entity linkage problem is performed offline, and consequently, even though a fast solution is preferable in general, the effectiveness is much more important, and here *QuadSky* significantly outperforms the baselines.

9.8 Comparison with Supervised Learning Techniques

In this section, we keep our *QuadSky* steps but replace the labeling of the pairs with a supervised learning technique. We decided to compare the *SkyEx-** family of algorithms with logistic regression [43], support vector machines (SVM) [44], decision trees [45], and Naive Bayes [46], which are supervised learning techniques commonly used in entity resolution problems [8, 10, 25, 33, 47]. We applied these methods on D_{full} pairs that are at most 30 meters apart (dataset description in Table 3). We experimented with training on 75% of D_{full} and testing on the remaining 25% with 4-fold cross validation (D_{full}-D_{full}), training on 75% of D_{sample} and testing on the remaining 25% with 4-fold cross validation (D_{sample}-D_{sample}), and training on D_{sample} and testing on D_{full} (D_{sample}-D_{full}). The results are presented in Table B.6. While logistic regression and SVM yield a slightly higher *F-measure* of 0.76 in D_{full}-D_{full}, our algorithms, which do not build their model on labeled data, have almost the same *F-measures* (0.74 for *SkyEx-F* and *SkyEx-D*) in D_{full}-D_{full}. For the manually labeled dataset in D_{sample}-D_{sample}, our algorithms perform the second best (*F-measure* of 0.84), after the decision trees. *SkyEx-F* and *SkyEx-D* outperform the logistic regression, SVM, and the Naive Bayes, which yield *F-measures* of 0.81, 0.81, and 0.72, respectively. Having a large training set as in D_{full}-D_{full} is unrealistic in most real cases. Thus, we tried a more realistic scenario, where one would prepare a small manually labeled training set, and then, test the trained model on the full data (D_{sample}-D_{full}). *In this (most realistic) case, SkyEx-F and SkyEx-D outperform all supervised methods by 0.03-0.05 in F-measure, showing the main weakness of supervised models, namely*

	D_{full} -D_{full}			D_{sample}-D_{sample}			D_{sample}-D_{full}		
Method	**Pr.**	**Rec.**	**F1**	**Pr.**	**Rec.**	**F1**	**Pr.**	**Rec.**	**F1**
Log. reg.	0.83	0.70	**0.76**	0.80	0.83	0.81	0.70	0.72	0.71
SVM	**0.88**	0.67	**0.76**	0.81	0.80	0.81	0.71	0.70	0.71
Dec. Tree	**0.88**	0.66	0.75	**0.93**	0.82	**0.87**	0.65	0.74	0.69
Naive B.	0.71	**0.77**	0.74	0.63	**0.85**	0.72	0.62	**0.77**	0.69
SkyEx-F	0.80	0.69	0.74	0.87	0.82	0.84	0.80	0.69	**0.74**
SkyEx-D	0.81	0.68	0.74	0.87	0.82	0.84	**0.81**	0.68	**0.74**

Table B.6: Comparison with supervised learning

that the D_{sample} model is not representative enough when applied to D_{full}.
In general, the spatial entity linkage problem suffers from the lack of labeled data [12–14]. Consequently, the applicability of supervised learning techniques is limited. On the contrary, *SkyEx-D* is completely unsupervised and can still achieve results similar to a supervised technique. If the labeled data is present, note that supervised learning techniques build the model on the labeled data, whereas *SkyEx-F* and *SkyEx-FES* use the labels only to tune the threshold because the construction of the skylines is independent of the labels. For this reason, *in contrast to supervised learning, SkyEx-F, and SkyEx-FES do not require a big and representative training set, do not struggle with class imbalance, do not overfit the data, and their dimensionality is minimal (one skyline versus high-dimensional data).*

9.9 Comparison of SkyEx-D to Clustering Techniques

In Sect. 7.2, we claimed that clustering techniques would not manage to create two clusters: one for the positive-class pairs and one for the negative-class pairs. In this section, we will replace *SkyEx-D* with common clustering techniques and evaluate the formed clusters. We are comparing to distance-based clustering (k-means [48] and k-medoids [49]), hierarchial clustering [50] (agglomerative), and density-based clustering (DBSCAN [51]). The results are presented in Table B.7. For k-means and k-medoids, we specified the number of clusters as 2. For the hierarchical clustering, we cut the dendrogram to create two clusters. For DBSCAN, we tried several values of minimum points and ϵ to form either two clusters, or one cluster and noise points. We report the version with the noise points in the table because it yields better results. For the labeling, we tried both versions (labeling cluster 1 as positive and the rest as negative and vice-versa) and report the best version in the table. Distance-based clustering yields the best results, having the highest recall but with a very low precision of 0.28 in D_{full}, and the second-best (after *SkyEx-D*) *F-measure* of 0.74 in D_{sample}. Hierarchical clustering achieves higher precision than distance-based but with a very low recall of 0.11 in D_{full}, while the results are reversed to a high recall of 0.91 and a low precision of 0.23 in D_{sample}. For DBSCAN, the best values were achieved when we labeled the cluster as negative, and the noise points as positive, resulting in a recall of 1.0, but a very low precision of 0.23 in D_{full} and 0.26 in D_{sample}. This means that the positive-class pairs are not dense enough to form a cluster. Our *SkyEx-D* focuses more on the distance between the classes rather than within the classes, and thus outperforms clustering.

10 Conclusions and Future Work

Location-based sources provide rich details and semantics about spatial entities. However, identifying which pairs of spatial entities refer to the same physical entity is a challenging problem. In this paper, we addressed the problem of spatial entity linkage across multiple location-based sources. We proposed *QuadSky*, an approach that consists of a spatial blocking technique *QuadFlex*, pairwise comparisons with suitable similarity metrics for each attribute, a skyline-based ranking algorithm *SkyRank*, and the *SkyEx-** family of algorithms for classifying the pairs. *QuadFlex* arranges the spatial entities into spatial blocks with a low execution time (4-8 times less than FNN [42]) and without missing relevant comparisons (99.99% of FNN comparisons). *SkyEx-F* achieves 0.84 p and 0.84 r on a manually labeled dataset and 0.87 p and 0.6 r on an automatically labeled dataset. We provided a theoretical guarantee to prune 73% of the skyline explorations in *SkyEx-F* with the novel *SkyEx-FES* without any loss of *F-measure*. Our fully unsupervised *SkyEx-D* finds k_d very close to the optimal k_f (an *F-measure* loss of just 0.01). The *SkyEx-** family of algorithms outperforms the existing baselines in terms of *F-measure* and approximates the results of a supervised learning solution without the need of a labeled dataset, while *SkyEx-D* yields far better results than the clustering techniques. *SkyEx-F* and *SkyEx-D* are already available in the R `skyex` package [52], together with other functions for entity linkage. In future work, we aim to study different blocking techniques that combine several attributes and extend our *SkyEx-** family of algorithms to general (non-spatial) entity resolution problems.

References

[1] D. G. Brizan and A. U. Tansel, "A. survey of entity resolution and record linkage methodologies," *Communications of the IIMA*, vol. 6, no. 3, p. 5, 2006.

	D_{full}			D_{sample}		
Method	**Prec.**	**Rec.**	**F1**	**Prec.**	**Rec.**	**F1**
K-means	0.28	0.96	0.44	0.62	0.92	0.74
K-medoids	0.28	0.96	0.44	0.62	0.92	0.74
Hierarchial	0.62	0.11	0.19	0.23	0.91	0.36
DBSCAN	0.23	**1.00**	0.37	0.26	**1.00**	0.42
SkyEx-D	**0.81**	0.68	**0.74**	**0.87**	0.82	**0.84**

Table B.7: Comparing SkyEx-D to clustering techniques

[2] K. Shu, S. Wang, J. Tang, R. Zafarani, and H. Liu, "User identity linkage across online social networks: A review," *Acm Sigkdd Explorations Newsletter*, vol. 18, no. 2, pp. 5–17, 2017.

[3] C. T. Yui, L. J. Liang, W. J. Soon, and W. Husain, "A survey on data integration in bioinformatics," in *International Conference on Informatics Engineering and Information Science*. Springer, 2011, pp. 16–28.

[4] D. Firmani, B. Saha, and D. Srivastava, "Online entity resolution using an oracle," *Proceedings of the VLDB Endowment*, vol. 9, no. 5, pp. 384–395, 2016.

[5] R. Maskat, N. W. Paton, and S. M. Embury, "Pay-as-you-go configuration of entity resolution," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXIX*. Springer, 2016, pp. 40–65.

[6] J. Efremova, B. Ranjbar-Sahraei, H. Rahmani, F. A. Oliehoek, T. Calders, K. Tuyls, and G. Weiss, "Multi-source entity resolution for genealogical data," in *Population reconstruction*. Springer, 2015, pp. 129–154.

[7] M. Edwards, S. Wattam, P. Rayson, and A. Rashid, "Sampling labelled profile data for identity resolution," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 540–547.

[8] O. Goga, H. Lei, S. H. K. Parthasarathi, G. Friedland, R. Sommer, and R. Teixeira, "Exploiting innocuous activity for correlating users across sites," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 447–458.

[9] A. Panchenko, D. Babaev, and S. Obiedkov, "Large-scale parallel matching of social network profiles," in *International Conference on Analysis of Images, Social Networks and Texts*. Springer, 2015, pp. 275–285.

[10] S. Isaj, N. B. Seghouani, and G. Quercini, "Profile reconciliation through dynamic activities across social networks," in *International Conference on Advanced Information Systems Engineering*. Springer, 2019, pp. 126–141.

[11] R. Karam, F. Favetta, R. Kilany, and R. Laurini, "Integration of similar location based services proposed by several providers," in *International Conference on Networked Digital Technologies*. Springer, 2010, pp. 136–144.

[12] B. Berjawi, E. Chesneau, F. Duchateau, F. Favetta, C. Cunty, M. Miquel, and R. Laurini, "Representing uncertainty in visual integration." in *DMS*, 2014, pp. 365–371.

[13] A. Morana, T. Morel, B. Berjawi, and F. Duchateau, "Geobench: a geospatial integration tool for building a spatial entity matching benchmark," in *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2014, pp. 533–536.

[14] S. Isaj, E. Zimányi, and T. B. Pedersen, "Multi-source spatial entity linkage," in *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*, 2019, pp. 1–10.

[15] X. L. Dong and D. Srivastava, "Big data integration," in *2013 IEEE 29th international conference on data engineering (ICDE)*. IEEE, 2013, pp. 1245–1248.

[16] P. Christen, T. Churches, and M. Hegland, "Febrl–a parallel open source data linkage system," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2004, pp. 638–647.

[17] O. Peled, M. Fire, L. Rokach, and Y. Elovici, "Entity matching in online social networks," in *2013 International Conference on Social Computing*. IEEE, 2013, pp. 339–344.

[18] G. Quercini, N. Bennacer, M. Ghufran, and C. N. Jipmo, "Liaison: reconciliation of individuals profiles across social networks," in *Advances in Knowledge Discovery and Management*. Springer, 2017, pp. 229–253.

[19] M. Magnani and D. Montesi, "A survey on uncertainty management in data integration," *Journal of Data and Information Quality (JDIQ)*, vol. 2, no. 1, pp. 1–33, 2010.

[20] M. Schäfers and U. W. Lipeck, "Simmatching: adaptable road network matching for efficient and scalable spatial data integration," in *Proceedings of the 1st ACM SIGSPATIAL PhD Workshop*, 2014, pp. 1–5.

[21] R. Abdalla, "Geospatial data integration," in *Introduction to Geospatial Information and Communication Technology (GeoICT)*. Springer, 2016, pp. 105–124.

[22] P. Tabarro, J. Pouliot, R. Fortier, and L.-M. Losier, "A webgis to support gpr 3d data acquisition: A first step for the integration of underground utility networks in 3d city models," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, p. 43, 2017.

[23] S. Balley, C. Parent, and S. Spaccapietra, "Modelling geographic data with multiple representations," *International Journal of Geographical Information Science*, vol. 18, no. 4, pp. 327–352, 2004.

[24] V. Walter and D. Fritsch, "Matching spatial data sets: a statistical approach," *International Journal of geographical information science*, vol. 13, no. 5, pp. 445–473, 1999.

[25] V. Sehgal, L. Getoor, and P. D. Viechnicki, "Entity resolution in geospatial data integration," in *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, 2006, pp. 83–90.

[26] A.-M. O. Raimond and S. Mustière, "Data matching–a matter of belief," in *Headway in spatial data handling*. Springer, 2008, pp. 501–519.

[27] G. Papadakis, E. Ioannou, C. Niederée, T. Palpanas, and W. Nejdl, "Beyond 100 million entities: large-scale blocking-based resolution for heterogeneous data," in *Proceedings of the fifth ACM international conference on Web search and data mining*, 2012, pp. 53–62.

[28] G. Papadakis, J. Svirsky, A. Gal, and T. Palpanas, "Comparative analysis of approximate blocking techniques for entity resolution," *Proceedings of the VLDB Endowment*, vol. 9, no. 9, pp. 684–695, 2016.

[29] H. Samet, "The quadtree and related hierarchical data structures," *ACM Computing Surveys (CSUR)*, vol. 16, no. 2, pp. 187–260, 1984.

[30] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8. Soviet Union, 1966, pp. 707–710.

[31] C. Fellbaum, "Wordnet," in *Theory and applications of ontology: computer applications*. Springer, 2010, pp. 231–243.

[32] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *ACL*, 1994.

[33] H. Köpcke, A. Thor, and E. Rahm, "Evaluation of entity resolution approaches on real-world match problems," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 484–493, 2010.

[34] Y. Censor, "Pareto optimality in multiobjective problems," *Applied Mathematics and Optimization*, vol. 4, no. 1, pp. 41–59, 1977.

[35] G. M. Weiss and H. Hirsh, "Learning to predict extremely rare events," in *AAAI workshop on learning from imbalanced data sets*. AAAI Press Austin, 2000, pp. 64–68.

[36] D. A. Cieslak and N. V. Chawla, "Learning decision trees for unbalanced data," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2008, pp. 241–256.

[37] J. Zhang, E. Bloedorn, L. Rosen, and D. Venese, "Learning rules from highly unbalanced data sets," in *Fourth IEEE International Conference on Data Mining (ICDM'04)*. IEEE, 2004, pp. 571–574.

[38] M. Gordon and M. Kochen, "Recall-precision trade-off: A derivation," *Journal of the American Society for Information Science*, vol. 40, no. 3, pp. 145–151, 1989.

[39] P. Roocks, "Relational and algebraic calculi for database preferences," Ph.D. dissertation, Universität Augsburg, 2016.

[40] M. Endres, P. Roocks, and W. Kießling, "Scalagon: an efficient skyline algorithm for all seasons," in *International Conference on Database Systems for Advanced Applications*. Springer, 2015, pp. 292–308.

[41] S. Isaj and T. B. Pedersen, "Seed-driven geo-social data extraction," in *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*, 2019, pp. 11–20.

[42] J. L. Bentley, D. F. Stanat, and E. H. Williams Jr, "The complexity of finding fixed-radius near neighbors," *Information processing letters*, vol. 6, no. 6, pp. 209–212, 1977.

[43] D. W. Hosmer and S. Lemeshow, "S.(1989) applied logistic regression," *John Wiley&Sons*.

[44] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[45] W. A. Belson, "Matching and prediction on the principle of biological classification," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 8, no. 2, pp. 65–75, 1959.

[46] T. Bayes, "Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s," *Philos Trans R Soc Lond*, 1763.

[47] G.-w. You, S.-w. Hwang, Z. Nie, and J.-R. Wen, "Socialsearch: enhancing entity search with social network matching," in *Proceedings of the 14th International Conference on Extending Database Technology*, 2011, pp. 515–519.

[48] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[49] L. Kaufman and P. Rousseeuw, "Clustering by means of medoids," 1987.

[50] T. Hastie, R. Tibshirani, and J. Friedman, "Hierarchical clustering," *EoSL*, 2009.

[51] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[52] S. Isaj and T. B. Pedersen, "skyex: an r package for entity linkage," in *International Conference on Extending Database Technology*, 2020, pp. 587–590.

Paper C

skyex: an R Package for Entity Linkage

Suela Isaj and Torben Bach Pedersen

The paper has been published in the
23rd International Conference on Extending Database Technology (EDBT), March 30-April 2, 2020, Copenhagen, Denmark, ISBN: 978-3-89318-083-7

Abstract

As the data is becoming bigger, more heterogeneous, and originating from different sources, the availability of the same information in different forms leads to various entity linkage problems. We demonstrate our `skyex` *package, an R package that supports all three steps of entity linkage: blocking, pairwise comparison, and labeling. Thus, the user can solve the whole process using* `skyex`*, but not necessarily; the* `skyex` *modules are independent, meaning that the user can easily integrate them with other packages or even other environments. Additionally, we are the first to provide the implementation of two skyline-based algorithms (*SkyEx-F *and* SkyEx-D*) that can label the compared pairs without the need for weights, scoring functions, etc.* `skyex` *supports the typical workflow of entity linkage, using minimalist, user-friendly function calls.*

1 Introduction

The *entity linkage* problem, sometimes called *data matching*, *entity resolution*, *duplicate detection*, *reconciliation*, etc., detects different records that belong to the same entity. Even though the process varies in different domains, the main steps are the same: blocking, pairwise comparison, and labeling the pairs (Fig.C.1). The entity linkage process starts with a set of entities that might contain duplicates. First, a blocking method is used to group entities that show a certain level of similarity and are of interest to compare further. Then, the pairwise comparison step compares the entities in the same blocks, e.g., using similarity metrics of the attributes of the entities or comparing the structure of their connections. Finally, the labeling step decides whether a pair of candidates belongs to the same entity or not. The entity linkage process results in a set of labeled pairs.

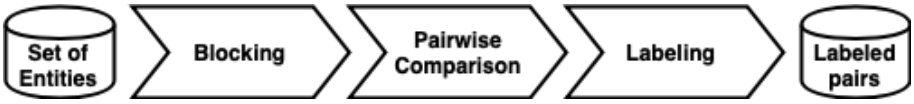


Fig. C.1: The entity linkage process

We present an R package, `skyex`, that supports all three steps of the entity linkage problem. In the labeling step, we provide the novel *SkyEx-F* and *SkyEx-D* algorithms in [1, 2]. The R language is in the top five languages of data science, and even more importantly, R is the second most used software in data science scientific papers, corresponding to 50,000 articles [1]. Moreover, R is used by different industries besides academia, such as healthcare, government, insurance, etc., where entity resolution is a common obstacle [2]. The current entity linkage tools [3–9] offer rule-based solutions with blocking and comparison functions [3, 4], crowdsourcing solutions [5, 9], or machine learning solutions [6–8]. In contrast to all the current tools, we contribute with a Labeling module that implements two novel algorithms (*SkyEx-F* and *SkyEx-D*) [1, 2], which can label the pairs without the need of weights, scoring functions, or exhaustive experiments. In order to support the full entity linkage workflow, we provide functions to perform blocking based on text and spatial attributes, and we offer a module for textual, spatial, semantic pairwise comparison. Similarly to [6], we support analysis and visualization functions that assist in the interpretation of the results and assessing the quality of the labeling. Analogously to [8] that uses Python, `skyex` uses the R ecosystem and can be easily integrated with other packages, in contrast to the current standalone tools. Finally, we demonstrate the different scenarios

[1] http://r4stats.com/articles/popularity/
[2] https://stackoverflow.blog/2017/10/10/impressive-growth-r/

that can be supported by our tool using three real-world datasets. Overall, `skyex` solves the entity linkage problem with minimal effort and background knowledge.
The remainder of the paper continues with the functionalities covered by our `skyex` package in Section 2, a description of our on-site demonstration in Section 3, and finally, concluding in Section 4.

2 SkyEx package functionalities

The `skyex` package is composed of 17 functions corresponding to four main modules: Blocking, Pairwise Comparison, Labeling, and Analysis and Visualization. The workflow of using `skyex` is illustrated in Fig. C.2. The user starts with a dataframe (a common data type for storing tables in R) of entities. In order to illustrate the workflow and our functions, we will use a real-world dataset of spatial entities extracted as in [10] and used in the experiments of [1]. The dataset contains spatial entities in the North Denmark region, originating from four sources, Google Places, Yelp, Krak (online yellow pages in Denmark, www.krak.dk), and Foursquare. We also introduce the running example of six records of entities (`entities`) from this dataset in Fig. C.2, which are identified by an ID, by geographic coordinates latitude and longitude, categories that explain the type of spatial entity, and the address.

Blocking module After loading the data, we can use a blocking technique (textual or spatial) from the Blocking module. The textual blocking is executed by the `textual.blocking` function, choosing a similarity metric among *levenshtein*, *cosine*, *jaccard*, *jaro-winker*, and *qgram*, and setting a maximal distance allowed. For example, `textual.blocking` on the attribute "name" with *levenshtein* and maximal distance 4 will group the entities with names "Bilhuset Biersted A/S" and "Bilhuset Biersted" (from `entities` in Fig. C.2). Note that `textual.blocking` is accurate but time-consuming. Alternatively, `prefix.blocking` and `suffix.blocking` produce faster results. Besides, in some domains, e.g., for species names, these methods can be more relevant than textual blocking. For spatial entities, being spatially close is often a better indicator of block quality than the name. For example, two records with the same name, e.g., Fakta supermarkets in different cities, are two different entities. `spatial.blocking` creates blocks of entities that are at most *max_distance* meters apart. The code snippets for these blocking methods are as follows:

```
#Textual blocking using levenshtein distance and max_distance=4
blocks<-textual.blocking(data=entities, column = "name",
        method = "levenshtein", max_distance = 4)
#Prefix blocking for the first 4 characters
```

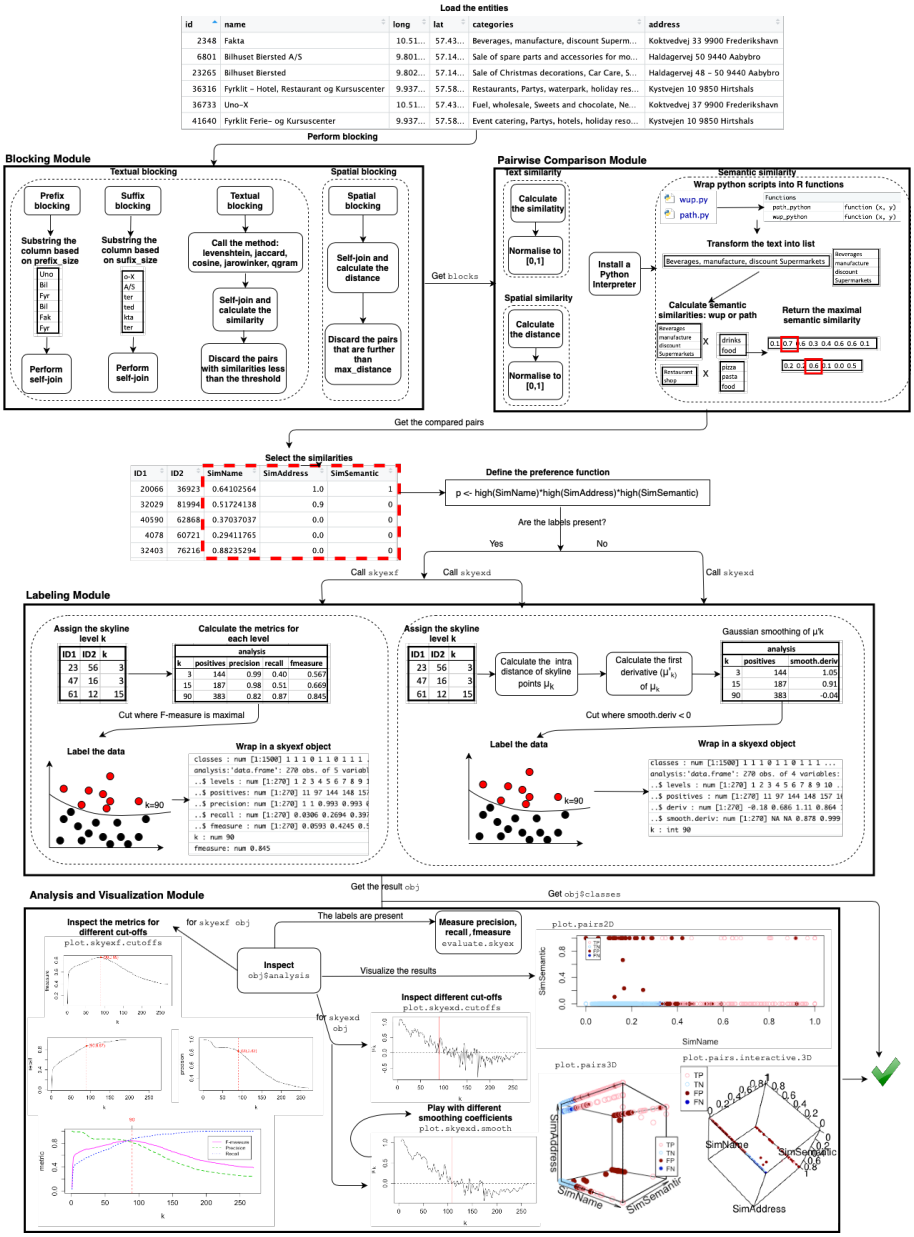


Fig. C.2: skyex workflow

```
blocks<-prefix.blocking(data=entities, column = "name", prefix_size = 4)
#Spatial blocking for entities at most 50 m apart
blocks<-spatial.blocking(data=entities, longitude = "long",
        latitude = "lat", max_distance = 50)
```

Pairwise Comparison module The Blocking module outputs a dataframe of pairs, which saves the user from the task of having to create the pairs from each block. The Pairwise Comparison module offers three functions that compare text syntactically and semantically, as well as spatial attributes. Moreover, all three functions output normalized values, which can be directly used in the Labeling module. `text.similarity` calculates the similarity of the pairs based on a text attribute using similarity metrics such as *levenshtein*, *cosine*, *jaccard*, *jaro-winker*. *levenshtein* similarity is calculated using the formula in [1, 11] in order to return a normalized value. `spatial.similarity` also requires a maximal distance for the normalization. For example, for a $max\_distance = 70$, "Uno-X" and "Fakta" will have a similarity of 0.0, because their distance of 83 meters is beyond the threshold. In the case of Bilhuset Biersted A/S and Bilhuset Biersted, this distance is 63 meters, which translates to a similarity of 0.09.
Regarding the semantic similarity, our work in [1] uses the Wu&Palmer metric from Wordnet. There exists a `wordnet` library in R, but it does not provide the metrics. Moreover, Wu&Palmer needs the whole path of both words that need to be compared, which in R, it could be resolved only through recursive calls. Through experimentation, this implementation turned out to be non-efficient. Thus, we include two Python scripts in the `skyex` package for two different metrics in Wordnet. These scripts are wrapped in R functions; thus, the user only needs to have a Python interpreter installed and give its path to the R function. The code for the pairwise comparisons is as follows:

```
#Text similarity using cosine
blocks$SimName<-text.similarity(data=blocks, method = "cosine",
                column1 = "name.x", column2 = "name.y")
#Spatial similarity with max_distance=70
blocks$SimSpatial<-spatial.similarity(data=blocks, lat1 = "lat.x",
                   long1 = "long.x", lat2 = "lat.y", long2 = "long.y",
                   max_distance = 70)
#Semantic similarity with Wu&Palmer
blocks$SimSemantic<-semantic.similarity(data=blocks,
                    column1 = "categories.x", column2 = "categories.y",
                    pythonpath = "/Users/..", method = "wup" )
```

Labeling module After the pairs are compared, the user can decide which similarities should go into the labeling process. Usually, he would select those similarities that are likely to indicate a match, e.g., the similarity of the names of the entities. We will consider the similarity of the name "SimName", the similarity of the address "SimAddress", and the semantic similarity of the

categories "SimSemantic" as in [1]. Then, the user decides on the preference function for the Pareto Optimality calculations. In our case, we prefer a high value for each similarity. Depending on the availability of the labels, the user can choose between running `skyexf` or `skyexd`, corresponding to the threshold-based *SkyEx-F*, or to the fully unsupervised *SkyEx-D*, respectively [2]. *SkyEx-F* finds that skyline level k that separates best the classes and maximizes the F-measure. It starts with assigning the skyline to all the points and then checking different cut-offs while measuring precision, recall, and f-measure. Finally, it labels the data, and the `skyexf obj` is returned, containing the classes, an analysis data frame, the proposed cut-off k, and the corresponding f-measure.
For unlabeled data, *SkyEx-D* finds the skyline level k where the mean distance of the points in the positive class starts to drop, meaning that we are entering the denser area of the negative class. It starts by assigning the corresponding skyline to each point; then, calculating the cumulative mean distance in the positive class and its first derivative; later, finding where the first derivative becomes negative for the first time. Finally, *SkyEx-D* labels the data and wraps the classes, the analysis data frame, and the proposed cut-off k in a `skyexd obj`. Detailed explanations about both algorithms can be found in [2]. Our `skyex` package hides all the details above from the user, meaning that the processes inside the dotted line boxes (Fig. C.2) are performed simply by the `skyexf` and `skyexd` function calls. The script for running both algorithms, storing the results of each labeling algorithm in separate objects, and attaching the predicted classes to the dataset is as follows:

```
#Define the preference
p<-high(SimName)*high(SimSemantic)*high(SimAddress)
#Call SkyEx-F algorithm and store the result in f.obj
f.obj<-skyexf(data=blocks, p=p, label="Class",posclass=1, negclass=0)
#Call SkyEx-D algorithm and store the result in d.obj
d.obj<-skyexd(data=blocks, p=p, simlist=c("SimName", "SimSemantic",
        "SimAddress"), posclass=1, negclass=0, smooth.coefficient=5)
blocks$fpred<-f.obj$classes
blocks$dpred<-d.obj$classes
```

We thus provide a labeling procedure that can be used with only two lines of code: defining the preference and calling the labeling function. However, for a more knowledgeable user, we offer the possibility to do analysis and visualize the results through the Analysis and Visualization module.

Analysis and Visualization module To illustrate the analysis of the labeling, we will use the 1500 manually-labeled pairs in [1]. Additionally, this dataset is also available in our package under the name `pairsManual` and can be loaded simply by `data(pairsManual)`. The Analysis and Visualization module needs the output of the Labeling module as input, which is a `skyexd` or `skyexf` object. The raw analysis can be accessed simply by calling the dataframe `analysis` from *obj* (inspect `obj$analysis` in Fig. C.2). In

the case of a `skyexf` object, `analysis` contains all the cut-offs, the size of the positive class, precision, recall, and f-measure. In order to facilitate the exploration of `analysis`, the user can call `plot.skyexf.cutoffs`, which produces graphs that monitor the evolution of the metrics when passing to the deeper skylines (see Fig. C.2). `plot.skyexf.cutoffs` by default plots the f-measure. However, it is possible to plot the precision and the recall separately, and also all metrics together. The code snippets for plotting the f-measure (first two), the precision, the recall, and all the metrics are as follows:

```
plot.skyexf.cutoffs(f.obj)
plot.skyexf.cutoffs(f.obj, "fmeasure")
plot.skyexf.cutoffs(f.obj, "precision")
plot.skyexf.cutoffs(f.obj, "recall")
plot.skyexf.cutoffs(f.obj, "all")
```

The resulting plots from the above script on `pairsManual` are shown in Fig. C.2 in the Analysis and Visualization module. Understandably, precision is high in the first skylines because it is very likely that the pairs in the first skylines that we label as positives are actual positives, but it degrades while moving in deeper cut-offs. On the contrary, recall is always increasing, the more we label as positive, the more likely it is to find an actual positive. The F-measure gives the trade-off between both metrics. All graphs show the suggested cut-off by `f.obj` in the red dotted line. However, the user can explore different trade-offs for his problem. In that case, plotting all metrics in a graph (the last script) gives a better overview.
In the case of a `skyexd` object, `analysis` keeps the cut-offs, the size of the positive class, the first derivative, and the smoothed values. Similarly, `plot.skyexd.cutoffs` aids exploring the raw `analysis` by plotting the smoothed first derivative function for each cut-off. If the plot looks too "smoothed" or too "raw", it is possible to play with different smoothing coefficients without having to re-run `skyexd` again by calling `plot.skyexd.smooth`. (see the code below). Fig. C.2 shows the analysis of `skyexd`, which was run with `smooth.coefficient=5`, and also the results of `plot.skyexd.smooth(d.obj, 10)`. The higher the smoothing coefficient, the higher the cut-off k, since smoother values push the cut-off towards deeper skylines.

```
#Plot the first derivative and the current cut-off k
plot.skyexd.cutoffs(d.obj)
#Smooth the first derivative with 10
plot.skyexd.smooth(d.obj, 10)
```

`evaluate.skyex` can also be called as in the code below, to measure precision, recall, and f-measure when the labels are available. The values of these metrics will be printed in the console.

```
evaluate.skyex(prediction=d.obj$classes, labels=data$Class, posclass = 1)
```

Additionally, we offer user-friendly functions to plot the data and the *obj* results. We offer 2D plots, 3D plots, and interactive 3D plots, where the user

can play and move the dimensions while looking at the data. The color of the points reflects if the pair is a true positive TP (an actual positive labeled as positive), a true negative TN (an actual negative labeled as negative), a false positive FP (an actual negative labeled as positive), and a false negative FN (an actual positive labeled as negative). The user can decide to change the colors of the points based on his preference. The code for these plots is as follows:

```
#Plot 2D using SimName and SimSemantic
plot.pairs2D(data=data, sim1="SimName", sim2="SimSemantic",
             prediction=f.obj$classes, labels=data$Class, posclass = 1)
#Plot 3D using SimName, SimSemantic, and SimAddress
plot.pairs3D(data=data, sim1="SimName", sim2="SimSemantic",sim3="SimAddress",
             prediction=f.obj$classes, labels=data$Class, posclass = 1)
#Plot 3D interactive plot using SimName, SimSemantic, and SimAddress
plot.pairs.interactive.3D(data=data, sim1="SimName", sim2="SimSemantic",
             sim3="SimAddress", prediction=f.obj$classes,
             labels=data$Class, posclass = 1)
```

Fig. C.2 shows the results of `pairsManual` with the three types of plots. These graphs can also be considered as an analysis since they show the problems with labeling and where to locate them. For example, it is noticeable that we have a bigger problem with the false positives then with the false negatives, thus if precision is fundamental to the domain, we could go back to the analysis and evaluation module and consider a smaller k for the cut-off. The interactive 3D plot offers a better view of the data points since it is possible to move and rotate the graph.

Summary The workflow of `skyex` supports typical entity linkage tasks, from blocking to evaluating the quality of the labels. The Blocking, Pairwise Comparison, and Labeling modules are completely independent, which means that the user can decide to perform his own methods and still be able to connect to the workflow of `skyex`. The labeling task can be as simple as just calling two lines of code to get the classes and as detailed as performing analysis, playing with the parameters, visualizing the labels, and highlighting the errors, etc. Moreover, the user can always go back, choosing new similarities and new preferences until the results are satisfactory. The `skyex` package is dependent on `rPref`, `dplyr`, `fields`, `rgl`, `plot3D`, `smoother`, `fuzzyjoin`, `stringr`, `stringdist`, `geosphere`, `reticulate`, and `pracma` which support some basic functionalities in our functions. `skyexf` and `skyexd`) scale relatively well for an R environment; e.g. they run in less than a minute for 50,000 pairs, less than 15 minutes for 150,000 pairs, and around 1 hour for 300,000 pairs.

3 Demonstration Overview

In the on-site demonstration, the user can download `skyex`[3], which is publicly available in GitHub, by following the README instructions, or use our pre-installed R environment. We will provide three datasets: `entities` (2814 spatial entities in the North Denmark region with an ID, name, categories, and address) [1], `restaurants`[4] (a collection of 864 restaurant records with name, address, city, and type), and `pairsManual` (1500 labeled pairs with pre-compared similarities of the name, address, and categories) [1]. Additionally, we have published a full video[5] demonstrating our functionalities for all three datasets, and a short video[6] for the `restaurants` dataset. We will provide example scripts, which the user can adapt based on his preference. The user will start with different blocking techniques on `entities` and `restaurants`, discussing with us what would be a good blocking technique for this dataset. Afterwards, he can play with different similarity metrics and different thresholds for the pairwise comparison. Later, the user can decide either to continue with the dataset of pairs he created so far from `entities` and `restaurants`, or move to the pre-compared `pairsManual` and play with the labeling parameters. The user can try both algorithms and will be guided by us through the Analysis and Visualization module. He can try the visualizations (including the interactive plotting) in order to detect problems with the labeling. Finally, he can discuss with us the applicability of the method across domains and possibilities for improvement.

4 Conclusions and Future Work

We introduced the `skyex` package, a user-friendly R package that supports all three steps of the entity linkage process. We demonstrated the functions of `skyex` with scripts and sample data, and supported the full workflow of the user. We showed that our Labeling module could solve the labeling problem with only two lines of code, but at the same time, offer the possibility for deeper analysis for the knowledgeable user. As future work, we intend to work on the scalability of our tool for big data, as well as on a similar package in Python.

[3] https://github.com/suelai/skyex
[4] source: https://www.cs.utexas.edu/users/ml/riddle/data.html
[5] https://youtu.be/TdxVsUtKRjw
[6] https://youtu.be/Zn8FOOh_xwA

References

[1] S. Isaj, E. Zimányi, and T. B. Pedersen, "Multi-source spatial entity linkage," in *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*, 2019, pp. 1–10.

[2] S. Isaj, T. B. Pedersen, and E. Zimányi, "Multi-source spatial entity linkage," 2019.

[3] A. Elmagarmid, I. F. Ilyas, M. Ouzzani, J.-A. Quiané-Ruiz, N. Tang, and S. Yin, "Nadeef/er: Generic and interactive entity resolution," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 1071–1074.

[4] L. Kolb, A. Thor, and E. Rahm, "Dedoop: Efficient deduplication with hadoop," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 1878–1881, 2012.

[5] X. Ke, M. Teo, A. Khan, and V. K. Yalavarthi, "A demonstration of perc: probabilistic entity resolution with crowd errors," *Proceedings of the VLDB Endowment*, vol. 11, no. 12, pp. 1922–1925, 2018.

[6] A. Ebaid, S. Thirumuruganathan, W. G. Aref, A. Elmagarmid, and M. Ouzzani, "Explainer: entity resolution explanations," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 2000–2003.

[7] E. C. Dragut, M. Ouzzani, A. K. Elmagarmid, W. G. Aref *et al.*, "Orlf: A flexible framework for online record linkage and fusion," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 2016, pp. 1378–1381.

[8] P. Konda, S. Das, P. Suganthan GC, A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. Naughton *et al.*, "Magellan: Toward building entity matching management systems," *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 1197–1208, 2016.

[9] Y. Govind, E. Paulson, P. Nagarajan, G. Paul Suganthan, A. Doan, Y. Park, G. Fung, D. Conathan, M. Carter, and M. Sun, "Cloudmatcher: a hands-off cloud/crowd service for entity matching," 2018.

[10] S. Isaj and T. B. Pedersen, "Seed-driven geo-social data extraction," in *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*, 2019, pp. 11–20.

[11] S. Isaj, N. B. Seghouani, and G. Quercini, "Profile reconciliation through dynamic activities across social networks," in *International Conference on Advanced Information Systems Engineering*. Springer, 2019, pp. 126–141.

Paper D

Explainable and Robust Skyline-Based Spatial Entity Linkage on Tiny Training Data

Suela Isaj, Vassilis Kaffes, Torben Bach Pedersen, and Giorgos Giannopoulos

The paper in under submission

Abstract

The ease of publishing data on the web has contributed to larger and more diverse types of data. Entities that refer to a physical place and are characterized by a location and different attributes are named spatial entities*. Even though the availability of spatial entity data from multiple sources seems attractive, there is unavoidable redundancy and ambiguity. We address the problem of spatial entity linkage with SkylineExplore-Trained (*SkyEx-T*), a skyline-based algorithm that can label an entity pair as being the same physical entity or not. We introduce LinkGeoMl-eXtended (*LGM-X*), a meta-similarity function which computes similarity features specifically tailored for spatial entities. The skylines of* SkyEx-T *are created using a preference function, which ranks the pairs based on the likelihood of referring to the same entity. We propose deriving the preference function using a tiny training set (down to 0.05% of the dataset). Additionally, we provide a theoretical guarantee for the cut-off that can best separate the classes, and we show experimentally that it results in a near-optimal F-measure (on average only 2% loss).* SkyEx-T *yields an F-measure of 0.71-0.73 and 0.81-0.83 in two different real-world datasets, respectively. In summary,* SkyEx-T *offers state-of-the-art machine learning level performance using very little training data, high model explainability, and high robustness in real deployment (the model will not need parameter tuning or further configuration to be deployed).*

1 Introduction

Web data is growing continuously in size and heterogeneity, providing rich and diverse information about different entities. Nowadays, we can rely on different sources for the same information, making the process of obtaining data more transparent and source-independent. Some of this web data is connected to a location, like a geographical point, or an address, thus, referring to a physical spatial entity. A spatial entity, apart from pointing to a location, is also characterized by an identity, which is constructed by a set of attributes, such as the name, the type, the phone number, reviews of people, etc. For example, “Restaurant Ambiance" is a spatial entity, located at (55.6,7.9), with the phone +4511111111 and the tags “restaurant" and “cosy". Different and independent sources can provide information about the same spatial entity. However, this independence of the sources and the ease of publishing data has also brought redundancy and sometimes even ambiguity. Several records of the same spatial entity might exist in the same source or across different sources. Continuing the previous example, another source might contain similar information about “Restaurant Ambiance", but now located at (55.7,7.8), with a different phone +4522222222 and the tags “restaurant" and “classy". The process of deriving which records belong to the same real-world physical spatial entity is known as *spatial entity linkage*.
The problem of spatial entity linkage can aid the research fields that work with spatial entities such as geo-recommender systems, influential locations, trajectory pattern mining, etc. Linked spatial entities contain a richer and integrated representation of the real entity. Additionally, spatial entity linkage can also improve the quality of the data for the industries that use spatial entities as the main input for their activities, such as marketing companies, tax offices, etc. Hence, spatial entity linkage is a problem that involves multiple stakeholders in different fields. Unfortunately, research on spatial entity linkage has not progressed at the same rate as the entity resolution research. Many papers focus on entity linkage of human entities [1–6], while spatial entity linkage has only been superficially addressed, usually contributing only a tool and leaving the decisions to be made by the user, rather than providing an automatic algorithm [7–9]. The works give little details about the accuracy of their methods and make arbitrary decisions to choose thresholds and scoring functions. The latest works in the field [10–14] provide solid progress on the accuracy of the models. The solutions in [10, 11] use preference functions to form skylines and rank the pairs based on the likelihood of referring to the same physical entity, but they choose them in an ad-hoc fashion, while [12–14] rely fully on machine learning without offering explainability for the end-user. Instead, combining two different approaches (skyline-based and traditional machine learning), we propose a skyline-based

algorithm *SkyEx-T* trained on a tiny training set of domain-specific similarity features *LGM-X*, which have proven to increase the interlinking accuracy in similarity comparison based [12] and classification based [13] settings. We make the following contributions:

- We propose a skyline-based algorithm *SkyEx-T* that can learn a preference function and a cut-off using a tiny training set (down to 0.05% of the data).
- We provide a theoretical guarantee for the cut-off selection in *SkyEx-T*, and we show experimentally that the selected cut-off yields a near-optimal F-measure.
- We propose the domain-specific LinkGeoMl-eXtended (*LGM-X*) similarity-based features that accommodate the specifics of spatial entities.
- We offer high model explainability and robustness for immediate deployment while providing machine-learning-level accuracy.
- We evaluate our approach on two real-world datasets, each originating from four and two different sources, respectively.

We study the related work in Section 2, formulate the problem of spatial entity linkage in Section 3, describe the main components of our solution, introduce *LGM-X*, and propose *SkyEx-T* in Section 4, provide the experimental results in Section 5, and finally, conclude and point to future work in Section 6.

2 Related Work

The entity linkage problem is addressed widely in several papers, across various domains [1, 5, 6, 15–17]. There are several terms related to this problem, such as data integration, entity resolution, deduplication, etc. The entity linkage process consists of three main steps: *blocking*, *pairwise comparison*, and *pair labeling*. Our work is focused on pairwise comparison and the pair labeling.

Entity resolution. A vast amount of research in entity resolution focuses on humans. Textual attributes [18], profile photos [4], network of friends [15, 19], publications [3, 20, 21], social media posts [5, 6] are used to distinguish the pairs of entities that might refer to the same individual. The spatial entity resolution problem shares only the basics with the human entity resolution problem in terms of comparing attributes such as name, keywords, etc., while the recent advances of the entity resolution research focus entirely on human networks, human activities, temporal and spatial traces of humans, etc. Consequently, they cannot apply to spatial entities.

Spatial entity blocking. There are several blocking techniques that use textual attributes to group similar entities [22–25], but few considering spatial entities and their geographical coordinates [8–11]. In most of the works, the spatial entities are grouped using user-defined distance thresholds. For example, in [7], the spatial entities that are at most 5 m apart are grouped. The work in [9] defines a threshold based on the type of spatial entities, e.g., 50 m for restaurants, but 500 m for parks. Following the same idea, the authors of [10, 11] propose an algorithm, *QuadFlex*, that is inspired from a quadtree and can adapt the radius based on the density of the spatial entities in an area. In this way, the algorithm will be more restrictive for spatial entities in the city center, setting a smaller radius than spatial entities in the countryside. Spatial blocking techniques are not in the scope of this work, so we use the state of the art *QuadFlex* approach [10, 11] to create our pairs.

Spatial entity pairwise comparison. There is research on spatial data integration, focusing on integrating spatial objects, which, in contrast to spatial entities, are identified only by their geographic coordinates and sometimes their shape [26–29]. Spatial entities might share the same spatial object while still belonging to different entities, e.g., a restaurant and a hairdresser are located in the same geographical point, but on different floors of the same building. As a result, we need to compare the attribute values of spatial entities in order to decide whether they belong to the same physical entity or not. The more similar two spatial entities are, the more likely they refer to the same physical spatial entity. To measure the similarity, the attribute values of the entities are compared, and similarity features are calculated. For name similarities, one could use Levenshtein, Jaccard, Cosine [5, 6], or more advanced metrics such as a Soft-TFIDF measure combined with Levenshtein [30]. Santos et al. [31] present a thorough overview of 13 different string similarity functions and perform an extended comparison of these functions on the accuracy achieved on the name attributes of entities. Additionally, the computed similarity scores are used as training features in state-of-the-art supervised machine learning algorithms for classification. However, these metrics are general and not designed explicitly for spatial entities. The metrics proposed in [32, 33] are specifically designed for the name attributes of spatial entities that mostly correspond to variations of the procedures used for generic name matching. Davis et al. [32] present a hybrid, three-stages method, i.e., the DAS similarity measure, that combines features from token-based and edit-based approaches. In [33], a four-stage process is proposed that takes into account accentuation and other language-specific aspects of spatial entities names. Recchia et al. [34] evaluates the set of algorithms presented in [35] on place names listed in the GEOnet Names Server, that contains romanized entity names from 11 different countries. Based

on their study, no similarity measure achieves the highest accuracy in all datasets. A different approach is followed in the problem of business places deduplication by Delvi et al. [36]. In the proposed solution, authors identify words of higher significance (core terms) that use to build a name model. This model is properly combined with a spatial context model using an unsupervised learning algorithm. Although the above methods attempt to incorporate the specifics of spatial entities name attributes, they do not achieve state-of-the-art accuracy results in the entity matching problem.
Deep learning methods for the name entity matching problem are also being proposed in the literature. Santos et al. [37] present a method, based on Siamese RNNs, for addressing the task of entity matching. Their method yields better accuracy results than traditional classifiers on similarity-based training features. Alexis et al. propose an Attention-based network model and a hybrid scheme model that combines individual machine and deep learning approaches and achieves the highest accuracy reported results on the Geonames toponym dataset. However, these methods require large amounts of data to properly engineer and train deep network model architectures, which does not apply in our setting.
In [13, 14] the authors propose the LGM-Sim meta-similarity function that incorporates domain knowledge for the toponym interlinking problem. They demonstrate that applying their method on top of several baseline similarity functions improves the interlinking effectiveness by a large extent. Moreover, they utilize training features derived from LGM-Sim within classifiers, showing a further increase in accuracy.

Spatial entity pair labeling. After having pairwise similarities and features, the pairs need to be labeled as positive if they belong to the same physical entity or as negative, otherwise. Sehgal et al. [38] stand between spatial data integration and spatial entity integration because their entities have names, geographical coordinates, and types but they refer to spatial objects representing landscapes (lakes, hills, rivers, etc.). They used supervised learning to calculate the similarity of the spatial entity/object types and another classifier that learns the weights of each similarity in a training set and assigns the class accordingly. Similarly, the works in [7, 8] assign weights to the similarity scores of the attributes: Berjawi et al. [8] arbitrarily constructs a similarity score based on an average of the attribute similarities, while Karam et al. [7] use the coefficient $\frac{2}{3}$ for the name, the coordinates and the type of the entity and the coefficient $\frac{1}{3}$ for the website, the address and the phone number. Morana et al. [9] uses textual and semantic attributes and takes the labeling decision based on belief theory [39]. Isaj et al. [10, 11, 40] propose a skyline-based labeling method that instead of assigning weights, construct a preference function using the Pareto dominance concept and separate the classes

by defining the number of skylines. In [10, 40], the number of skylines, which was a parameter in [11], is set using an unsupervised algorithm. However, the authors do not provide a feature selection method, and the preference function is chosen arbitrarily. Besides, all the research mentioned above uses general similarity metrics, sometimes with slight tuning [38], which are not very effective on spatial entities.

Summary. While we can observe significant advances in the field of entity resolution, the spatial entity linkage has mostly been handled as a non-spatial entity linkage problem and has only been explicitly addressed in very few papers. The research in spatial entity pairwise comparison has proposed effective spatial metrics for measuring the similarity between entities [33, 36, 37] and the current state of the art [12–14] achieves high accuracy without the need to train deep networks. In contrast to the current spatial entity linkage works, our proposed solution *SkyEx-T* uses *LGM-X* features, which extend the state of the art LGM-Sim features used for toponym interlinking [13, 14]. Moreover, for spatial pair labeling, instead of using weights like in [7, 8, 38], we use the Pareto operator. However, differently from [10, 11, 40], besides the Pareto operator, we can also prioritize features, and the preference function is not chosen arbitrarily; instead, we propose an algorithm for selecting the preference selection and the cut-off using very little labeled data.

3 Problem Definition

In this section, we will introduce some definitions and formulate the problem of Spatial Entity Pair Labeling. First, let us define a spatial entity:

Definition D.1. *A spatial entity s is a uniquely identified entity, located in a geographical point with coordinates $\langle long, lat \rangle$, and described by a set of attributes $A = \{a_1, a_2, ..., a_n\}$ with values $\{v^{a_1}, v^{a_2}, ..., v^{a_n}\}$.*

> **Example D.1**
> An example of a spatial entity is a restaurant located in a point $\langle 55.7, 8.9 \rangle$ with the name "Restaurant Amelie", with a phone number, an address, and categories such as {food, coffee, cosy}. In this case, the name, the phone number, the address, and the categories are the attributes a_1, a_2, a_3, and a_4, respectively. Similarly, the values of the attributes are $\{v^{a_1}$ = "Restaurant Amelie", v^{a_2} = +45111111, and v^{a_3} = {food, coffee, cosy}.

To discover which spatial entities belong to the same physical entity, we need to compare them. The spatial entities can be compared to each other in an

exhaustive way (Cartesian product), or, to reduce the number of comparisons, they can be grouped in blocks based on one or more attributes [11, 22–25]. Within the blocks, the spatial entities are compared pairwise with respect to their attribute values. From the pairwise comparison, we obtain *features*, defined as follows:

Definition D.2. *A feature* $X_i^{a_n}$ $(A \times A \mapsto [0,1])$ *is a function that computes the similarity between the values* $v_i^{a_n}$ *and* $v_j^{a_n}$ *of the attribute* a_n *for the spatial entities* s_i *and* s_j*, respectively.*

Note that from the definition, there can be more than one computed feature for the same attribute, for example:

Example D.2
Let us consider two spatial entities s_1 and s_2 named "Amelie" and "Ami", respectively. Let a_1 be the name attribute. Thus, we have $v_1^{a_1}$ = "Amelie" and $v_2^{a_1}$ = "Ami". We can compute several similarities on the name attribute, e.g., Jaccard, cosine, Jaro-Winker. Let $X_1^{a_1}$, $X_2^{a_1}$, and $X_3^{a_1}$ be the Jaccard, cosine, and Jaro-Winker computed similarities of $v_1^{a_1}$ and $v_2^{a_1}$. Therefore, we have $X_1^{a_1} = 0.6$, $X_2^{a_1} = 0.6123724$, and $X_3^{a_1} = 0.8333333$.

For two spatial entities, we will compute the features and obtain a featured pair of spatial entities:

Definition D.3. *A featured pair of spatial entities* $\langle s_i, s_j, X \rangle$ *is formed by two spatial entities* s_i *and* s_j *and their features* $X = \{X_1^{a_1}, X_2^{a_2}, ..., X_N^{a_n}\}$*, where the features are computed for their respective pairs of attribute values* $\{\langle v_i^{a_1}, v_j^{a_1} \rangle, \langle v_i^{a_2}, v_j^{a_2} \rangle, ..., \langle v_i^{a_n}, v_j^{a_n} \rangle\}$ *for the attributes* $\{a_1, a_2, ..., a_n\}$*.*

Note that we need methods to compare the different attributes of the spatial entities. In the next sections, we will discuss similarity metrics and machine-learning techniques to compare the spatial entities (the features). The intuition behind the comparison is that the more similar two spatial entities are regarding their features, the more likely it is that they refer to the same physical spatial entity. Thus, we want to assign a *label* to each pair; a positive label if the entities of the pair have high values of similarities and are the same physical spatial entity, and a negative label if they are different physical spatial entities.

Definition D.4. *A labeled pair of spatial entities* $\langle s_i, s_j, C_{ij} \rangle$ *is formed by two spatial entities* s_i *and* s_j *and their class* C_{ij}*. The class* C_{ij} *is either positive* ($C_{ij} = 1$) *when the pair* $\langle s_i, s_j \rangle$ *is likely to refer to the same physical spatial entity, or negative* ($C_{ij} = 0$)*, otherwise.*

Problem definition: *The Spatial Entity Pair Labeling problem is a classification problem that for a given pair of spatial entities $\langle s_i, s_j \rangle$, aims to determine whether s_i and s_j refer to the same physical spatial entity or not.*
In other words, the Spatial Entity Pair Labeling problem aims to transform unlabeled pairs of spatial entities into labeled ones. In the next sections, we will introduce our solution that computes the features of the pairs and labels them based on the similarity of the spatial entities.

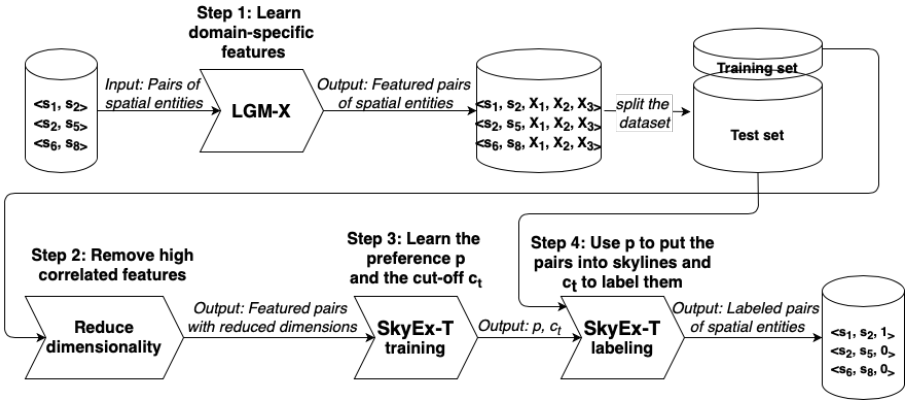


Fig. D.1: Overview of SkyEx-T with LGM-X

4 SkyEx-T with LGM-X

In this section, we introduce our approach to solving the spatial entity linkage problem, *SkyEx-T* with *LGM-X* features, which uses domain-specific features for expressing the similarity between spatial entities and a skyline-based algorithm trained on a tiny dataset for labeling the pairs of spatial entities.

4.1 Overview of SkyEx-T with LGM-X

We will briefly introduce the main components and the workflow of our proposed approach (Fig. D.1). We start with a set of pairs of spatial entities. We apply *LGM-X* (detailed in Section 5.1) to produce domain-specific features, which capture better the characteristics of spatial entities (Step 1 in Fig. D.1). *LGM-X* produces featured pairs of spatial entities. We split this output into a small training set and the rest as a test set. We use the training set for training the skyline-based algorithm *SkyEx-T* (detailed in Section 5.2). We start by reducing the dimensionality (Step 2 in Fig. D.1 and detailed in Section 5.2.1). This step will output featured pairs but with fewer features. Then, in Step 3, we learn the preference function p (Sections 5.2.2 and 5.2.3) and the

cut-off c_t (Section 5.2.4) by training *SkyEx-T*, which are needed to label the pairs. Finally, in Step 4, we use the preference function p to put the pairs in the test set into skylines, and the cut-off c_t to separate the ranked pairs and assign them into a class (Section 5.2.5). This final step produces the labeled pairs.

4.2 LGM-X

In this section, we present the *LGM-X* (inter-Linking Geo-spatial entities using Machine learning - eXtended), our proposed training features to feed the *SkyEx-T* algorithm for better capturing and exploiting the different attributes of spatial entities and their geographical coordinates. In the following subsections, we present a summary of the LGM-Sim [13] algorithm, a meta-similarity function that aims to capture meaningful entity characteristics and incorporate the specifics of spatial entities attributes related to name and address. Consequently, we discuss the training features we introduce, i.e. features derived from the proposed meta-similarity and additional entity specific features, that enhance the process of similarity matching.

4.2.1 LGM-Sim Overview

The LGM-Sim algorithm aims at properly splitting the strings of compared attributes of the spatial entities into discrete lists of terms, with each list containing terms of different semantics. It takes as input the name attributes of the two strings which are pre-processed in order to remove accents and punctuation marks. Further, a separate process decides whether the terms within the two strings should be alphanumerically sorted. Afterwards, the algorithm initiates the process of splitting the initial strings into three separate lists of terms each, i.e., (i) two *base* lists, (ii) two *mismatch* lists and (iii) two *frequent terms* lists. It firsts identifies frequent terms within the two strings and moves them to the respective lists, i.e., the *frequent terms* lists. This way, we isolate the least significant terms that are the least important in determining whether the two strings refer to the same entity or not. The other two types of lists will contain terms that potentially are of higher significance in the problem of spatial entity linkage. Specifically, the *base* lists will contain terms from the two strings that (loosely) match to each other and the *mismatch* lists will contain the rest of the terms that failed to match to the terms from the other string. Finally, for each type of list, individual similarity scores are computed which are differently weighted in order to produce the final similarity score of the two spatial entities. All the parameters of LGM-Sim, i.e., comparison thresholds and individual score weighting, are automatically learned on a training dataset. Additionally, the set of frequent terms are automatically gathered by the corpus of spatial entities contained in this training dataset.

Feature Type \ Attribute	Name	Address		Coords
		name	no	
Basic similarities	14	14		
Sorted similarities	13	13		
LGM-Sim based sims.	13	13		
Individual sim. scores	3	3		
Numerical			1	
Spatial				1

Table D.1: The considered LGM-X training features specialized for spatial entities

4.2.2 LGM-X Features

In this work, we adopt the set of training features presented in [13, 31] and enrich it with entity-specific defined features, that concern numerical and spatial aspects of the compared entities. Specifically, we adopt the features presented in Santos et al. [31] where a combinations of several generic similarity measures, like Damerau-Levenshtein, Jaro-Winkler, Jaccard N-Grams etc., are encoded as features for classifiers. Moreover, we generate a corresponding set of features by applying LGM-Sim on top of all similarity measures presented in [31]. Further, an “intermediate" set of training features are considered, again corresponding to the similarity measures presented in [31]. However, in this case, we apply only the sorting function of LGM-Sim, which decides whether to sort the terms of each string alphanumerically or not, before computing the similarity score of the entity names. Additionally, we consider the three individual similarity scores computed on the three types of individual lists that LGM-Sim splits the two entity names into. Note that the above set of features are applied to the name and address (only the text part of it) attributes of spatial entities. In addition to these, we define entity-specific training features, that concern numerical and spatial aspects of the compared entities. Specifically, the numerical distance of the address numbers of the compared spatial entities form one integer feature, whereas the Euclidean distance of their geographical coordinates form one float feature. Eventually, we consider six groups of training features (Table D.1): (i) the *basic* similarity features as presented in [31]; (ii) the *sorted* similarity features; (iii) the *LGM-Sim* based similarity features; (iv) the individual scores on the split of the attributes produced by applying LGM-Sim based Damerau-Levenshtein similarity on respective attributes; (v) the numerical feature on address number; and (vi) the spatial feature on geographical coordinates.

4.3 SkyEx-T

In this section, we present *SkyEx-T* (Skyline Explore - Trained), our skyline-based algorithm which labels the pairs as positives when there is a high likelihood that they belong to the same spatial entity, and negative, otherwise.

In the following subsections, we will detail how *SkyEx-T* learns the preference function p in a small training set, selects a cut-off ratio c_t, and then ranks the data based on the preference p and separates the classes according to c_t.

4.3.1 Reducing the dimensionality

Each pair is described as a set of N *LGM-X* features $\{X_1^{a_1}, X_2^{a_1}, X_3^{a_2}, ..., X_N^{a_n}\}$ that are computed on the values of the attributes $\{a_1, a_2, ..., a_n\}$ as described in Section 5. To have a simpler notation, from now on, we will not use the superscript of the attribute in the features but will simply write $\{X_1, X_2, X_3, ..., X_N\}$. Some of these features are highly correlated, given that they measure the similarity of the same attributes, e.g., measuring Jaccard and cosine similarity on the names of the spatial entities. To remove highly correlated features, we use the *mutual information* (MI) [41] to detect the dependency between two variables. MI can detect different relationships, and it is not limited to linear correlation (like Pearson's correlation). MI measures the similarities of the joint distribution of the two variables. If $p_x(x)$ and $p_y(y)$ are the marginal probability density functions of variables x and y, respectively, and $p_{x,y}(x,y)$ is the joint probability function of both x and y, then the MI of x and y is defined as:

$$MI(x,y) = \int_x \int_y p_{x,y}(x,y) log \frac{p_{x,y}(x,y)}{p_x(x)p_y(y)} \tag{D.1}$$

After the features are pairwise compared using MI, we identify those pairs of features that are highly correlated. Then, we remove one from each of the highly correlated pairs; we choose to remove the feature with the largest mean correlation overall. From the remaining features, we need to build a model to predict the class of the pairs.

4.3.2 Preference functions

In contrast to traditional machine learning techniques that fit the training set to a model, we propose having a function that is expressed as a preference of feature values; the pairs that are ranked better with respect to the preference function are the pairs that are likely to refer to the same physical entity.

Definition D.5. *A preference function $p : P \mapsto P_k$ is a function that takes as input a set of pairs $P = \{\langle s_i, s_j \rangle\}$ and outputs the partially ranked (some pairs share the same rank) list of pairs $P_k = \{\langle s_i, s_j, k \rangle\}$ with respect to their feature values, where $k \in [1, m]$ is the rank of a pair $\langle s_i, s_j \rangle$, and m is the maximal rank.*

The preference function ranks the pairs from the most preferred pairs to the least preferred ones, meaning that the first rank pairs are more likely to be the same spatial entity than the rest. To show that a pair $\langle s_i, s_j \rangle$ is preferred

over another pair $\langle s'_i, s'_j \rangle$, we will use the symbol $\succ$, so $\langle s_i, s_j \rangle \succ \langle s'_i, s'_j \rangle$. The definition of the preference is based on the definition of the skylines. We use a definition similar to [10]:

Definition D.6. *A skyline of level k is a set of pairs* $Skyline(k) = \{\langle s_i, s_j \rangle\}$ *ranked in the* k^{th} *position according to the preference function p such that for each pair* $\langle s_i, s_j \rangle \in Skyline(k)$ *and for each pair* $\langle s'_i, s'_j \rangle \in Skyline(k')$ *where* $k' > k$, $\langle s_i, s_j \rangle \succ \langle s'_i, s'_j \rangle$.

According to the above definition, the pairs in $Skyline(k)$ are more likely to be the same spatial entity than the pairs in $Skyline(k')$, for $k > k'$. Let us now further detail the components of the preference functions: *the preferred feature direction* and *the preference operators*, defined as below:

Definition D.7. *The preferred feature direction* $d() : X \mapsto \mathcal{N}$ *is a function that takes as input the list of values of a feature X, orders them preferring either high values* $(high())$ *or low values* $(low())$*, and outputs the rank for each feature.*

$d()$ is a generic way to express the preferred feature direction, but in practice, we will use $high()$ or $low()$ based on the feature. Instead of coefficients, the preference function only specifies if we prefer high or low values of a feature. For example, for a pair to be labeled as positive (they are the same physical spatial entity), we would prefer a high Levenshtein similarity, so $high(X_{levenshtein})$. Note that we can construct a very basic preference function by using the preferred feature direction on only one feature. For example, $p = high(X_{levenshtein})$ will rank the pairs P based on their Levenshtein similarity and assign them to skylines. If we want to include more than one preferred feature direction in a preference function, we will need the preference operators. We introduce two types of preference operators: the *Pareto* operator and the *Priority* operator:

Definition D.8. *The Pareto operator* $\triangle$ *is a binary operator connecting two preferred feature directions according to the Pareto Optimality concept.*

The Pareto Optimality [42] concept is widely used in multi-criteria optimization problems. *A combination of* (x, y) *is Pareto optimal when there is no other combination that can increase x without decreasing y*. In our context, the Pareto operator prefers one solution over the rest if it is better in terms of at least one feature value, while for the remaining feature values it is either the same or better. Otherwise, the solutions are Pareto-Optimal, meaning that we have no preference of one over the other. Let us illustrate this concept with an example:

Example D.3
Let us consider the pair of spatial entities $\langle s_1, s_2 \rangle$, which are compared with regards to their name. We have two computed features that indicate the similarity of the name: X_1=0.7 and X_2=0.3. Since a high similarity is likely to indicate a match of the pairs, we prefer high values. Hence, the preferred feature direction is $high()$ for both features. Given that we have no information on whether we should prefer one feature over another, we choose to connect these features with the Pareto operator, so $high(X_1) \triangle high(X_2)$. This means that for another pair $\langle s_3, s_4 \rangle$ to be preferred over $\langle s_1, s_2 \rangle$ ($\langle s_3, s_4 \rangle \succ \langle s_1, s_2 \rangle$), $\langle s_3, s_4 \rangle$ has to be better at least in one of the features, while the other does not increase. For example $\{X_1 = 0.7, X_2 = 0.4\}$, $\{X_1 = 0.9, X_2 = 0.3\}$, $\{X_1 = 0.8, X_2 = 0.4\}$ would be combinations which are preferred over the $\{X_1 = 0.7, X_2 = 0.3\}$.

Besides the Pareto operator, we can also choose to prefer one feature over another. We introduce the *Priority* operator defined as follows:

Definition D.9. *The Priority operator $\triangleright$ is a binary operator connecting two preferred feature directions such that the feature direction on the left side of the operator is preferred over the one on the right side.*

The Priority operator expresses an order of the feature importance; for example, if the cosine similarity is more likely to detect matches than the Jaccard similarity, we prioritize those pairs that have high cosine similarity over those with high Jaccard similarity. Let us give an example:

Example D.4
Let us consider the pair of spatial entities $\langle s_1, s_2 \rangle$ of the previous example, with the similarities of the name: X_1=0.7 and X_2=0.3. Let us suppose that X_2 can detect the class better than $X1$. Even though high values of both features are preferred, we would be more interested in having high values of X_2. Thus, we express the preference as $high(X_2) \triangleright high(X_1)$. This means that for another pair $\langle s_3, s_4 \rangle$ to be preferred over $\langle s_1, s_2 \rangle$ ($\langle s_3, s_4 \rangle \succ \langle s_1, s_2 \rangle$), $\langle s_3, s_4 \rangle$ has to have a higher X_2, or the same X_2 but a better X_1. For example $\{X_1 = 0.8, X_2 = 0.3\}$ and $\{X_1 = 0.7, X_2 = 0.4\}$ would be combinations which are preferred over the $\{X_1 = 0.7, X_2 = 0.3\}$.

After defining the preferred feature directions and the preference operators, we can construct different preference functions. Let us illustrate this with an example:

Example D.5
Let us consider a pair of spatial entities $\langle s_1, s_2 \rangle$ of the previous example, with the similarities of the name: X_1=0.7 and X_2=0.3 and physical distance (in meters) from each other $X_3 = 10$. Since $X1$ and X_2 express the similarity of the name, we prefer a high value, so the preferred feature direction is $high(X_1)$ and $high(X_2)$. On the contrary, we prefer the spatial entities to be as close as possible to each other, thus, we prefer a low value on the distance, so $low(X_3)$. Let us suppose that we do not have any preferred order of X_1 and X_3, while X_2 can detect the class better than $X1$ and $X3$. In that case, we use the Pareto operator between X_1 and X_3 as in $high(X_1) \triangle low(X_3)$ and the Priority operator for X_2 as in $high(X_2) \triangleright (high(X_1) \triangle low(X_3))$. Hence, the preference function is $p = high(X_2) \triangleright (high(X_1) \triangle low(X_3))$.

When we have few features and little domain knowledge, we can select the preference function ourselves. However, when having many features, there are many possible ways to connect them with operators. In the next subsection, we introduce our algorithm *SkyEx-T* that automatically determines a suitable preference function.

4.3.3 Preference Training

The preference function in [10, 11] was chosen ad hoc, and the only operator was Pareto. To improve the process of preference selection, we offer expressiveness and agility by introducing a preference function that can be trained on a rather small training set, and thus automatically learn optimal preferences for each dataset. We propose the *SkyEx-T* algorithm (Skyline Explore - Trained) for preference training which further reduces the dimensionality and selects a preference function based on the importance of each feature for detecting the class. Then, $SkyEx-T$ ranks the pairs according to p and finds the cut-off ratio for separating the classes (this procedure will be detailed in the next section). The pseudocode of *SkyEx-T* is formalized in Algorithm D.1. To learn which remaining features are more suitable for the preference function, we calculate their correlation with the class C. C is 0 if the pairs are not a match and 1 otherwise. We want to find those features whose increase is usually associated with an increase in the class label (C changes from 0 to 1). We use Spearman's correlation because it is capable of detecting these monotonic relationships between variables, and it is not limited only to linear relationships.

$$\rho_{X_i} = \frac{cov(X_i, C)}{\sigma_{X_i} \sigma_C} \tag{D.2}$$

where $cov(X_i, C)$ is the covariance of the feature X_i and the class C, and σ_{X_i} and σ_C are the standard deviations of X_i and C, respectively.
After having all ρ_{X_i} values for each X_i, we select those X_i which have high values of ρ_{X_i}. Note that even high negative values are preferred: an increase in X_i leading to a decrease in C indicates that we can change the preferred direction of the feature from $high(X_i)$ to $low(X_i)$. To choose the features, we plot the absolute values of ρ_{X_i} in decreasing order and notice when ρ_{X_i} falls considerable. This will be graphically associated with an elbow in the curve. We denote the first elbow as ε_1 and the X_i features with $|\rho_{X_i}| \leq \varepsilon_1$ as X_{ε_1}. X_{ε_1} are showing the strongest monotonic relationships to C, so their increase is closely related to C. Given that these are the most important features, they will be prioritized over the rest of the features while using the Pareto operator amongst X_{ε_1} themselves, treating them equally. In order to make the function more accurate, we can refine it further using the less important features. For that, we will again search for the next elbow in the curve, denoted as ε_2. We denote with X_{ε_2} the features with $|\rho_{X_i}| \leq \varepsilon_2$ and $\rho_{X_i} > \varepsilon_1$. X_{ε_2} will be connected by the Pareto operator themselves, while they will be connected by the Priority operator with X_{ε_1}. To sum up, we have two groups of features: X_{ε_1} and X_{ε_2}. We use the Pareto operator among the X_{ε_1} and X_{ε_2} but the Priority operator for X_{ε_1} over X_{ε_2}. Let us illustrate this procedure with an example:

Example D.6
Let us suppose we have two spatial entities s_1 and s_2 with 10 features $\{X_1, X_2, ...X_{10}\}$. After computing ρ_{X_i} for each feature and ordering their absolute values in a descending order, we get these values {0.6, 0.56, 0.55, 0.54, 0.34, 0.33, 0.33, 0.32, 0.11, 0.06}, which are the $|\rho_{X_i}|$ values of $\{X_3, X_4, X_7, X_1, X_9, X_2, X_5, X_8, X_{10}, X_9\}$, respectively. Fig. D.2 shows $|\rho_{X_i}|$ values ordered from the highest to the lowest. It is possible to notice the two elbows in the curve, ε_1 and ε_2. Let us suppose that the preferred direction of each feature is $high()$. Then, the preference function will be as follows: $p = (high(X_3) \triangle high(X_4) \triangle high(X_7)) \triangleright (high(X_1) \triangle high(X_9) \triangle high(X_2) \triangle high(X_5))$.

The above procedure is formalized in lines 1-10 in Algorithm D.1. We first initialize R_X in line 1 to store all ρ_{X_i}. Then we calculate ρ_{X_i} for each feature and order R_X in lines 2-3. We find the elbows ε_1 and ε_2 in line 4. Then, we connect the features within X_{ε_1} and X_{ε_2} by the Pareto operator in lines 5-10. Finally, we prioritize X_{ε_1} over X_{ε_2} in line 11.

4.3.4 Skyline ranking and fixing the cut-off ratio

After having the preference function, we can apply it to the pairs and rank them accordingly. We use the skylines as in [11] to first find those pairs who are preferred the most when using preference p. Let us suppose that $Skyline(1)$ contains all the pairs $\{\langle s_i, s_j\rangle\}$ that, given preference p, $\langle s_i, s_j\rangle \succ \langle s'_i, s'_j\rangle$ for each $\langle s'_i, s'_j\rangle \in Skyline(k)$, where $k > 1$. Thus, all pairs in $Skyline(1)$ dominate the rest of the pairs. Then, we need to remove $Skyline(1)$ from the set of pairs. Applying the same preference, we choose the next set of most preferred pairs ($Skyline(2)$). We continue this procedure until there are no more pairs left.
While moving from one skyline to the next, we need to define which cut-off number of skylines best separates the classes. Given that the preferred pairs are likely to indicate a match, we will label as positive the first skylines up to the cut-off and the remaining as negative. Cutting too early will result in high precision, but low recall, while cutting too late will improve the recall, but at the cost of lower precision. Thus, we use a well-known balanced indicator, the F-measure ($F1 = \frac{2*precision*recall}{precision+recall}$). We measure the F-measure for each cut-off and fix the cut-off to k_l where the F-measure is maximized. We express this cut-off as a ratio of $\frac{k_l}{b}$ where b is the maximal level of skylines. Let us provide theoretical guarantees that a cut-off ratio found in a training set can satisfy the F-measure maximization in the whole dataset or the test set.

Theorem D.1. *Let us apply the preference function p on two random samples P_1 and P_2 from the pairs P. If P_1^k and P_2^k are the ranked pairs of P_1 and P_2 with regards to p, and $\mathcal{D}_1$ and $\mathcal{D}_2$ are the probability density of P_1^k and P_2^k, respectively, then $\mathcal{D}_1 \sim \mathcal{D}_2$.*

Proof. Based on the Central Limit Theorem [43], the sample distributions will respect the population distribution, around the same mean μ as the population P but with a smaller standard deviation proportional to the size of the

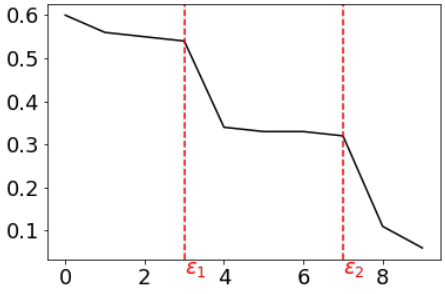


Fig. D.2: Finding ε_1 and ε_2

Algorithm D.1 SkyEx-T training

Input: A set of labeled pairs $P_t = \{\langle s_i, s_j, C_{ij}\rangle\}$
Output: A trained preference function p and cut-off c_t

$R_X \leftarrow \emptyset$
Calculate ρ_{X_i} for each X_i and add each $|\rho_{X_i}|$ to R_X
Order R_X in a descending order
Find ε_1 and ε_2 elbows in R_X
for each X_i that $|\rho_{X_i}| \leq \varepsilon_1$ **do**
  $p_1 = d(X_1) \triangle d(X_2) ... \triangle d(X_m)$
end for
for each X_i that $|\rho_{X_i}| > \varepsilon_1$ and $|\rho_{X_i}| \leq \varepsilon_2$ **do**
  $p_2 = d(X_m + 1) \triangle d(X_{m+2}) ... \triangle d(X_n)$
end for
$p = p_1 \triangleright p_2$
$P_k \leftarrow \emptyset$
$F \leftarrow \emptyset$
while $|P_k| < |P|$ **do**
  Find $Skyline(k) = \{\langle s_i, s_j\rangle\} \mid \forall \langle s', s''\rangle \in P_t - \{\langle s_i, s_j\rangle\}$, $\langle s_i, s_j\rangle \succ \langle s', s''\rangle\}$
  Remove $Skyline(k)$ from P and add it to P_k
  Label P_k as positive
  Calculate $F1(k)$ and add $F1(k)$ to F
  $P_t = P_t - Skyline(k)$

Find k_l such that $F1(k_l) = max(F1(k))\ \forall k \in \{1, |F|\}$
$c_t = \frac{\sum_{i=1}^{k_l} Skyline_i}{\sum_{i=1}^{max(k)} Skyline_i}$
return p, c_t

sample $\frac{\sigma}{\sqrt{n}}$. Let us denote by $f(k)$ the probability density function of the pairs with relation to their skyline level k. After applying p on P_1, P_2 and P, the new ranked pairs in P_1^k and P_2^k will respect their order of ranking in each sample. Note that $f(k)$ is deterministic. Let us suppose that we apply the preference p in P and we obtain the ranked pairs P^k. For every $\langle s_1, s_2\rangle$ and $\langle s_3, s_4\rangle$ in P^k, if $\langle s_1, s_2\rangle \succ \langle s_3, s_4\rangle$, $\langle s_1, s_2\rangle \in P_1$, and $\langle s_3, s_4\rangle \in P_1$, then we have that $\langle s_1, s_2\rangle \succ \langle s_3, s_4\rangle$ in P_1^k as well. Similarly, For every $\langle s_1, s_2\rangle$ and $\langle s_3, s_4\rangle$ in P^k, if $\langle s_1, s_2\rangle \succ \langle s_3, s_4\rangle$, $\langle s_1, s_2\rangle \in P_2$, and $\langle s_3, s_4\rangle \in P_2$, then we have that $\langle s_1, s_2\rangle \succ \langle s_3, s_4\rangle$ in P_2^k. This means that $f(k)$ will simply, in a deterministic way, reorder the observations of the samples, which already have similar distributions, strictly respecting the order in the in P^k. As a result, the new probability density distributions will be similar, so $\mathcal{D}_1 \sim \mathcal{D}_2$. □

Theorem D.2. *Let us apply the preference function p on two random samples P_1 and P_2 from P and denote by P_1^k and P_2^k their respective ranked pairs with regards to p. If k_1 is a cut-off in the probability density function $D_1 = \int_1^{b_1} f_1(k)dk$ (limits in $[1, b_1]$) of P_1^k such as the F-measure is maximal, then $\frac{k_1 b_2}{b_1}$ will also be a near-optimal cut-off for P_2^k with probability density function $D_2 = \int_1^{b_2} f_2(k)dk$ (limits in $[1, b_2]$).*

Proof. The probability density distribution for the ranked pairs P_1^k and P_2^k is $\int_1^{b_1} f_1(k)dk$ and $\int_1^{b_2} f_1(k)dk$, respectively, and the skyline levels lie in $[1, b_1]$ for P_1^k and $[1, b_2]$ for P_2^k. The work in [44] presents the preferred solutions of a decision maker in relation to the mean, variance, and third moments of a distribution. In other words, a preferred cut-off in a probability distribution $\mathcal{D}_1$ is a preferred solution for a probability distribution $\mathcal{D}_2$ as long as the mean, variance, and third moments are similar. When sampling P_1 and P_2 from P, we gain a sample of pairs with a mean μ as in P [43]. After ranking, the pairs near μ will be assigned in the k_{μ_1}-th skyline for the pairs P_1^k and in k_{μ_2} skyline for the pairs P_2^k. According to Theorem D.1, $\mathcal{D}_1 \sim \mathcal{D}_2$, so $k_{\mu_2} \simeq k_{\mu_1}$. The variances will be $\sigma_1 = \int_1^{b_1}(k - k_{\mu_1})^2 f(k)d(k)$ and $\sigma_2 = \int_1^{b_2}(k - k_{\mu_2})^2 f(k)d(k)$ for P_1 and P_2, respectively. Given that $k_{\mu_2} \simeq k_{\mu_1}$, we have that $\sigma_1 \simeq \sigma_2$. The third moment is an indicator of skewness and it is defined as: $m_1^3 = \frac{\int_1^{b_1}(y - k_{\mu_1})^3 dF(y)}{\sigma_1^3}$ and $m_2^3 = \frac{\int_1^{b_2}(y - k_{\mu_2})^3 dF(y)}{\sigma_2^3}$ for P_1 and P_2, respectively. Given that $k_{\mu_1} \simeq k_{\mu_2}$ and $\sigma_1 \simeq \sigma_2$, then $m_1^3 \simeq m_2^3$. The pairs in the first skylines have a high likelihood to belong to the same physical spatial entity and usually, they form a skewed long-tail distribution [10]. Let us suppose that $k = k_1$ is a cut-off in the probability density function $\int_1^{b_1} f(k)dk$ such that when labeling pairs in a skyline smaller than k_1 as positive and the rest as negative, the F-measure is maximized. In other words, k_1 is the preferred cut-off for the decision maker. According to the skewness condition in [44], a similar cut-off in the probability density distribution $\int_1^{b_2} f(k)dk$ of P_2 will yield a near-optimal F-measure value. In order for k_1 to be applicable to P_2, we need to express it as a ratio $\frac{k_1}{b_1}$ and adapt it to P_2 as in $\frac{k_1 b_2}{b_1}$. □

Note that this process is stochastic, meaning that it might not always find the optimal cut-off, but it will always find the near-optimal cut-off value (we also show this experimentally in Section 5). We will learn the cut-off ratio from a training set and then, given Theorem D.2, we will apply it to the test set. However, having the cut-off ratio $\frac{k_1}{b_1}$ requires that we rank all the pairs in the training set and moreover, to find $\frac{k_1 b_2}{b_1}$, we need b_2, which again will need the ranking of all the pairs in the test set. For a big dataset, this procedure will be time-consuming. Therefore, we use a simple derivative of Theorem D.2, formalized in Lemma 1, where instead of a cut-off in the number of skylines,

we use the percentage of the data belonging to the optimal level of skylines.

Lemma D.1. *If* $c_1 = \frac{\sum_{i=1}^{k_1} Skyline_i}{\sum_{i=1}^{b_1} Skyline_i}$ *is the ratio of the total pairs in* P_1^k *that belong to a skyline level of at most* k_1*, where* k_1 *is a cut-off in the probability density function* $D_1 = \int_1^{b_1} f_1(k)dk$ *(limits in* $[1, b_1]$*) of* P_1^k *such as the F-measure is maximal, then* $c_1 * |P_2^k|$ *is a near-optimal cut-off for* P_2^k *for maximizing the F-measure.*

Proof. The cut-off k_1 in P_1^k corresponds to the skyline level which can best separate the classes. Given that the probability density distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ of P_1 and P_2 are similar (Theorem D.1), for each pair $\langle s_i, s_j \rangle$ in P_1^k, the probability of belonging to the positive class is $P(k_{\langle s_i, s_j \rangle} \leq k_1) \sim \frac{\sum_{i=1}^{k_1} Skyline_i}{\sum_{i=1}^{b_1} Skyline_i}$. According to Theorem D.2, the cut-off ratio $\frac{k_1}{b_1}$ is near-optimal for P_2^k in order to best separate the classes. Let us denote with $c_2 = \frac{\sum_{i=1}^{k_2} Skyline_i}{\sum_{i=1}^{b_2} Skyline_i}$ the ratio of total pairs in P_2^k that corresponds to skyline levels up to k_2. Given that Theorem D.2 and $\mathcal{D}_1 \sim \mathcal{D}_2$, $\frac{\sum_{i=1}^{k_1} Skyline_i}{\sum_{i=1}^{b_1} Skyline_i} \sim \frac{\sum_{i=1}^{k_2} Skyline_i}{\sum_{i=1}^{b_2} Skyline_i}$. Thus, the ratio $c_1 \sim c_2$, so c_1 is the near-optimal for P_2^k. □

The skyline ranking is detailed in lines 12-19 in Algorithm D.1. We start with an empty set of explored skylines P_k in line 12. Note that in line 13 we are initializing F, which will keep the F-measure for each cut-off. We will need F later when we learn the cut-off c_t. While there are still pairs that are not ranked, we find the next skyline that satisfies the preference function (line 15) and remove it from the pairs P_t. We label all pairs in P_k as positive and calculate the F-measure for this cut-off in lines 17-18. Then, we remove the current skyline and continue the ranking. We find the best cut-off k_l that maximizes the F-measure in line 21. We express the cut-off as a data ratio (Lemma 1), which makes it applicable to any sample (line 22). Finally, we return the learned preference function p and the cut-off ratio c_t.

4.3.5 Classification of Pairs

In the previous subsections, we discussed the training of *SkyEx-T* (Algorithm D.1) that determines the preference function, ranks the pairs, and selects the cut-off. In this subsection, we will formalize *SkyEx-T* labeling algorithm for classifying the pairs (Algorithm D.2). We start with a set of unlabelled pairs P, a trained preference function p and cut-off ratio c_t from Algorithm D.1. We apply the preference function p on P. We rank the pairs in P according to the preference function p and assign them to skylines (1-5). Note that we do not rank all the pairs but only the pairs in the first skylines that constitute

the ratio c_t of all pairs, so a total of $c_t|P|$ pairs. These pairs are the ones that later will be labeled as positive (Lemma 1). When $c_t|P|$ pairs are ranked, we need to label them. We label all the ranked pairs in P_k as positive and the remaining unranked pairs as negative. Finally, we return the labeled P'.

Algorithm D.2 SkyEx-T labeling

Input: A set of unlabeled pairs $P = \{\langle s_i, s_j \rangle\}$, a trained preference function p and a cut-off ratio c_t
Output: A set of labeled pairs $P' = \{\langle s_i, s_j, C_{ij} \rangle\}$

$P_k \leftarrow \emptyset$
while $|P_k| < c_t|P|$ **do**
  Find $Skyline(k) = \{\langle s_i, s_j \rangle\} \mid \forall \langle s', s'' \rangle \in P - \{\langle s_i, s_j \rangle\}, \langle s_i, s_j \rangle \succ \langle s', s'' \rangle\}$ based on p
  Remove $Skyline(k)$ from P and add it to P_k
  $P = P - Skyline(k)$

Set $C_{ij} = 1$ (positive class) for all $\langle s_i, s_j \rangle \in P_k$ and add them to P'.
Set $C_{ij} = 0$ (negative class) for all $\langle s_i, s_j \rangle \in P - P_k$ and add them to P'.
return P'

5 Experimental Results

In this section, we will describe our datasets and then, report the results of our experiments with *SkyEx-T* and several machine learning techniques.

5.1 Dataset

We are using two datasets of spatial entities for the experiments: *North Denmark spatial entities* (North-DK) and the *Fodor's and Zagat's restaurants* (Restaurants).

North Denmark spatial entities **(North-DK)** contains spatial entities from Denmark, originating from four different sources, Google Places (GP), Foursquare (FSQ), Yelp, and Krak[1]. The spatial entities are retrieved by using the seed-driven approach described in [45]. We obtained 75,541 spatial entities, where 51.50% originate from GP, 46.22% from Krak, 0.03% from FSQ, and 2.23% from Yelp. The 75,541 retrieved spatial entities were grouped using the spatial blocking techniques in [11]. After the spatial blocking, we

[1]Krak (www.krak.dk) contains businesses, organizations, companies, etc. with their attributes such as name, location, address, phone, categories, etc. Krak is part of Eniro Danmark A / S., which is responsible for The Yellow Pages

obtained 777,446 pairs. To establish a ground truth, we apply the following rule; if the phone number or the website is the same, then we infer that the pair should refer to the same physical entity. By using this rule, we obtain 27,102 positive pairs (3.5% of the total pairs). In Table D.2, we observe that most of the positive pairs are from different sources, especially from Krak and Google Places (64.2% of the total positive pairs). Even though the spatial entities are uniquely identified within a source, we can still find duplicates within the same source. 28.7% of the total positive pairs originate from same-source pairs, especially in Krak and in Google Places.
Fodor's and Zagat's restaurants[2] **(Restaurants)** is a dataset of 864 restaurant entities, where 61.69% are extracted from Fodor's Travel[3] and 38.31% from Zagat website[4]. The entities are characterized by the name, the address, the city, the phone number, and the type of the spatial entity. Given that the geographical coordinates are missing in this dataset, we do not perform the spatial blocking but rather use all the pairs. We obtain 372,816 pairs, 112 out of which refer to the same spatial entity (0.03% of the total pairs). These 112 pairs are already given, so we did not have to compute the ground truth for this dataset. However, the phone number attribute has been used to derive the class, so we exclude the phone number from the pairwise comparison of attributes.

Source	Krak	GP	Yelp	FSQ
Krak	3,789	17,405	902	7
GP		3,546	968	13
Yelp			460	12
FSQ				0

Table D.2: The sources of the positive pairs

5.2 Comparing SkyEx-T to Machine Learning Solutions

In this section, we will compare the results of SkyEx-T to some popular machine learning techniques, used in spatial entity linkage [13]. We compare our results to Support Vector Machine (SVM) [46], Decision Trees [47], Random Forest [48], Extremely Randomized Trees (Extra Trees) [49], Extreme Gradient Boosted Trees (XGBoost) [50], Multi Layer Perceptron (MLP) [51]. To have a fair comparison, the machine learning techniques use the same *LGM-X* features as *SkyEx-T*, meaning that there was no further tuning to the features. We selected this experimental configuration in order to (a) assess the generalization of algorithms in different spatial entity datasets and (b) to

[2]https://www.cs.utexas.edu/users/ml/riddle/data.html
[3]https://www.fodors.com
[4]https://www.zagat.com

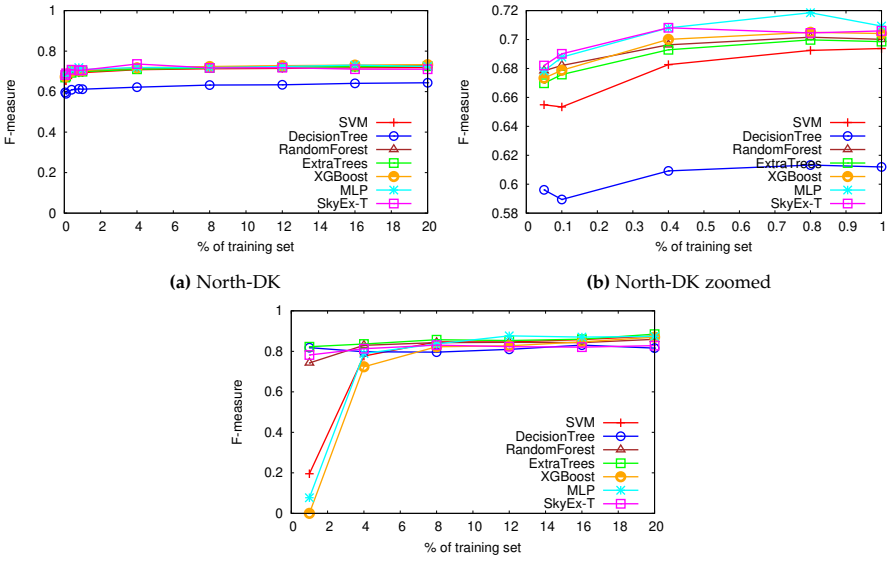
Training size	0.05%	0.10%	0.40%	0.80%	1%	4%	8%	12%	16%	20%	80%
						Precision					
SVM	**0.901**	**0.912**	**0.896**	**0.885**	**0.890**	**0.885**	0.880	0.882	0.882	0.880	0.876
DecisionTree	0.597	0.571	0.591	0.598	0.599	0.608	0.616	0.617	0.626	0.630	0.654
RandomForest	0.818	0.839	0.857	0.860	0.868	0.870	0.868	0.871	0.871	0.871	0.874
ExtraTrees	0.837	0.842	0.867	0.865	0.871	0.876	0.874	0.877	0.877	0.877	0.877
XGBoost	0.761	0.779	0.797	0.811	0.812	0.832	0.828	0.833	0.834	0.835	0.850
MLP	0.781	0.804	0.826	0.818	0.826	0.851	0.857	0.853	0.843	0.855	0.859
SkyEx-T	0.844	0.804	0.852	0.813	0.879	0.882	**0.913**	**0.899**	**0.908**	**0.915**	**0.907**
						Recall					
SVM	0.520	0.511	0.552	0.569	0.569	0.590	0.599	0.601	0.605	0.608	0.616
DecisionTree	**0.613**	0.614	**0.629**	0.631	0.625	**0.636**	**0.649**	**0.651**	**0.656**	**0.658**	**0.681**
RandomForest	0.587	0.576	0.587	0.593	0.587	0.607	0.617	0.621	0.624	0.628	0.655
ExtraTrees	0.569	0.566	0.577	0.588	0.584	0.597	0.607	0.612	0.616	0.619	0.645
XGBoost	0.611	0.602	0.624	0.625	0.622	0.631	0.643	0.647	0.650	0.653	0.666
MLP	0.609	0.602	0.624	**0.642**	**0.630**	0.624	0.621	0.630	0.646	0.630	0.631
SkyEx-T	0.591	**0.620**	0.607	0.626	0.594	0.632	0.591	0.599	0.584	0.581	0.606
						F-measure					
SVM	0.655	0.653	0.683	0.692	0.694	0.708	0.713	0.715	0.718	0.719	0.723
DecisionTree	0.596	0.589	0.609	0.613	0.612	0.622	0.632	0.634	0.641	0.644	0.667
RandomForest	0.678	0.682	0.696	0.702	0.700	0.715	0.721	0.725	0.727	0.730	**0.749**
ExtraTrees	0.670	0.676	0.693	0.700	0.699	0.710	0.717	0.721	0.723	0.726	0.744
XGBoost	0.673	0.679	0.700	0.705	0.704	0.717	**0.724**	**0.728**	0.731	**0.733**	0.747
MLP	0.678	0.688	**0.708**	**0.719**	**0.709**	0.719	0.719	0.724	**0.731**	0.724	0.727
SkyEx-T	**0.682**	**0.690**	**0.708**	0.705	0.706	**0.736**	0.717	0.718	0.711	0.711	0.727
						Difference from Max F-measure in %					
SVM	3.96%	5.36%	3.53%	3.76%	2.12%	3.80%	1.52%	1.79%	1.78%	1.91%	3.47%
DecisionTree	12.61%	14.64%	13.98%	14.74%	13.68%	15.49%	12.71%	12.91%	12.31%	12.14%	10.95%
RandomForest	0.59%	1.16%	1.69%	2.36%	1.27%	2.85%	0.41%	0.41%	0.55%	0.41%	0.00%
ExtraTrees	1.76%	2.03%	2.12%	2.64%	1.41%	3.53%	0.97%	0.96%	1.09%	0.95%	0.67%
XGBoost	1.32%	1.59%	1.13%	1.95%	0.71%	2.58%	0.00%	0.00%	0.00%	0.00%	0.27%
MLP	0.59%	0.29%	0.00%	0.00%	0.00%	2.31%	0.69%	0.55%	0.00%	1.23%	2.94%
SkyEx-T	0.00%	0.00%	0.00%	1.95%	0.42%	0.00%	0.97%	1.37%	2.74%	3.00%	2.94%

Table D.3: SkyEx-T versus Machine Learning on North-DK

retain the complexity of our models, e.g., depth of each tree in random forest search, comparable to *SkyEx-T*.

5.2.1 Model accuracy

We trained on 0.05%, 0.1%, 0.4%, 0.8%, 1%, 4%, 8%, 12%, 16%, and 20% of the North-DK dataset, and 1%, 4%, 8%, 12%, 16%, and 20% of the Restaurants dataset. We could not train on less than 1% of data in Restaurants because there are only 112 positive pairs, and we would then have 0 positive pairs. For *SkyEx-T*, we first removed the highly-correlated features and learned the correlation of the remaining features with the class. We used ε_1 and ε_2 to find X_{ε_1} and X_{ε_2} as explained in Algorithm D.1. We found the cut-off c_t and applied it to the rest of the data as in Algorithm D.1. For each training size, we repeated the experiment 10 times with disjoint training sets, and we report the averages of precision, recall and F-measure in Table D.3 and Table D.4, and the evolution of F-measure while increasing the training set in the



(a) North-DK

(b) North-DK zoomed

(c) Restaurants

Fig. D.3: SkyEx-T vs Maching Learning

Fig. D.3. Overall, *the choice of* LGM-X *features proved highly effective for all methods*, even on small training sets, which are atypical for machine learning, a finding also shown experimentally for *LGM-Sim* in [13]. *LGM-X* was used on an initial set of parameters and was still able to generalize and achieve good effectiveness on various dataset sizes and configurations, especially on training sizes of over 8%.

We can notice the advantage of *SkyEx-T* over the machine learning techniques for small training sets in North-DK. *SkyEx-T* has the highest F-measure for 0.05%, 0.1%, 0.4% (together with MLP), and 4% (see Fig: D.3b). Overall, *SkyEx-T* is in the top 3 best methods in terms of F-measure for North-DK in small training sets). For Restaurants, *SkyEx-T* is in the top 3 best methods for 1% and 4%. More importantly, *SkyEx-T* starts at a very high F-measure when most of the machine learning techniques fail (see Fig. D.3c). *SkyEx-T* has the highest precision of all methods for 8%, 12%, 16%, 20%, and 80% training in North-DK, and the highest recall for 0.1% in North-DK, and highest recall for 1%-8% in Restaurants. To sum up, even though the training of *SkyEx-T* is significantly lighter compared to machine learning techniques (we only use the label for checking correlations), it is still on average, on the top 3 best methods in terms of F-measure in small training sets.

If we observe the difference in percentage from the maximal F-measure for each training size (Table D.3), in North-DK, *SkyEx-T* differs on average by only 0.83% for training sets up to 20% of the data. In Restaurants (Table D.4),

Training size	1%	4%	8%	12%	16%	20%	80%
				Precision			
SVM	0.600	**0.988**	**0.944**	0.952	**0.941**	0.952	0.931
DecisionTree	0.847	0.872	0.821	0.846	0.870	0.862	0.893
RandomForest	**0.920**	0.905	0.911	0.929	0.918	0.940	0.923
ExtraTrees	0.910	0.917	0.920	0.932	0.925	0.942	**0.937**
XGBoost	0.000	0.979	0.915	**0.953**	0.928	**0.956**	0.930
MLP	0.311	0.885	0.904	0.910	0.911	0.939	0.884
SkyEx-T	0.773	0.826	0.851	0.839	0.855	0.863	0.849
				Recall			
SVM	0.132	0.644	0.770	0.764	0.790	0.812	0.855
DecisionTree	0.804	0.746	0.782	0.788	0.801	0.781	0.864
RandomForest	0.675	0.769	0.787	0.777	0.781	0.796	0.845
ExtraTrees	0.753	0.777	0.807	0.793	0.807	**0.839**	0.877
XGBoost	0.000	0.581	0.754	0.737	0.783	0.803	**0.895**
MLP	0.047	0.721	0.785	**0.851**	0.838	0.826	0.868
SkyEx-T	**0.830**	**0.820**	**0.824**	0.817	0.794	0.802	0.796
				F-measure			
SVM	0.196	0.777	0.847	0.846	0.858	0.875	0.889
DecisionTree	0.818	0.798	0.796	0.810	0.831	0.816	0.875
RandomForest	0.743	0.830	0.843	0.844	0.843	0.859	0.879
ExtraTrees	**0.823**	**0.836**	**0.857**	0.853	0.860	**0.885**	0.904
XGBoost	0.000	0.724	0.823	0.827	0.847	0.870	**0.910**
MLP	0.077	0.789	0.837	**0.877**	**0.870**	0.874	0.871
SkyEx-T	0.782	0.813	0.831	0.823	0.821	0.828	0.820
				Difference from Max F-measure in %			
SVM	76.23%	7.13%	1.23%	3.46%	1.41%	1.07%	2.30%
DecisionTree	0.56%	4.59%	7.16%	7.61%	4.56%	7.79%	3.84%
RandomForest	9.74%	0.77%	1.66%	3.67%	3.14%	2.88%	3.41%
ExtraTrees	0.00%	0.00%	0.00%	2.65%	1.18%	0.00%	0.66%
XGBoost	100.00%	13.46%	4.06%	5.61%	2.72%	1.70%	0.00%
MLP	90.62%	5.62%	2.40%	0.00%	0.00%	1.26%	4.22%
SkyEx-T	4.98%	2.78%	3.12%	6.09%	5.70%	6.41%	9.92%

Table D.4: SkyEx-T versus Machine Learning on Restaurants

SkyEx-T yields, on average, an F-measure of 4.85% less than the maximal on training sets up to 20%. Furthermore, on training sets less than 8%, this difference is only 3.63%, the second-best after ExtraTrees, while SVM, MLP and XGBoost yield an F-measure which is on average 30% lower than the maximal. Overall, *SkyEx-T performs better than the machine learning techniques for small training sets and as good as them for larger ones*. Additionally, it is important to note that there was no single best machine learning model that would give the best F-measure consistently for all the training sizes; apart from *SkyEx-T*, four different methods (RandomForest, ExtraTrees, XGBoost,

and MLP) competed for the highest F-measure; thus, there was no obvious winner.

5.2.2 Model explainability

Additionally to the results in terms of F-measure, we can compare the methods in terms of other criteria. Considering that spatial entities are important in many business applications, the models used for spatial entity linkage need to be explainable for the end-user, especially when these decisions have a high impact on the business. Moreover, “the right to explanation" is now officially a requirement of EU’s General Data Protection Regulation (GDPR) [52]. In the case of *SkyEx-T*, the preference model is human-readable and easily explainable. An example of a preference function is:
(high(SimName) $\triangle$ *high(LGM_baseScore)* $\triangle$ *high(SimAddress))* $\triangleright$ *(high(Sorted_Dice_bigrams)* $\triangle$ *high(Dice_bigrams)* $\triangle$ *high(Sorted_Soft_Jaccard)* $\triangle$ *high(LGM_Dice_bigrams))*.
It is simple to understand which s_1 and s_2 matched noticing what features the model prefers overall and which features over others. We can trace back the skyline ranking and understand why a specific label was assigned. As for the machine learning techniques, their complexity usually comes with a lack of explainability.
For the linear SVM, we can use the proposed probability estimates by Platt [53] to derive individual feature effects on the model outcome by perturbing one feature at a time through a range of values, whilst keeping the other features fixed. The Decision Trees model interpretation is rather simple, in theory, since each feature’s importance can be computed by measuring the amount of “impurity", i.e., the reduced variance or Gini index compared to the parent node. However, in practice, this is feasible only in cases where the tree depth is rather small. In our case, the depth of Decision Trees was small enough for the trees to be somewhat explainable for very small training sets in Restaurants: 2-3 for training sets of 0-10%, 4-6 for 10-20%, 7-9 for 80% but very large in North-DK: 28-34 for 0-10%, 30-42 for 10-20%, 40-45 for 80%, which makes the model unexplainable in practice. Nevertheless, both SVM and Decision Trees are outperformed, in terms of F-measure, in most scenarios by other algorithms, as presented in Tables D.3 and D.4. The interpretation gets even harder in tree-based models, e.g., Random Forest, Extra Trees and XGBoost, which consist of a large number of deep trees. Complex techniques, like Impurity Feature Importance or Conditional Permutation Feature Importance [54], are required to gain insights for the importance of the training features where each approach has problems and drawbacks and makes it difficult to apply in every case. Finally, the MLP (MultiLayer Perceptron) model is far from explainable since it is a type of a basic artificial neural network and, thus, it is treated as a black-box in most cases. To sum

up, all the machine learning techniques failed in terms of model explainability on both datasets while *SkyEx-T* successfully demonstrated very high explainability.

5.2.3 Model configuration and robustness

SkyEx-T has a very light configuration, meaning no coefficients or layered structures. This guarantees that firstly, *SkyEx-T cannot potentially overfit the training set*, and secondly, *it requires no tuning at all*. On the contrary, the machine-learning techniques with *LGM-X* are composed of parts that require a heavy fine-tuning pre-processing step. Each machine learning model needs hyperparameter tuning to select the optimal model architecture for the examined problem. Additionally, the core meta-similarity function LGM-X requires an additional search for optimal values in a set of parameters, like thresholds for sorting/splitting and assigned weights, applied to the name parts that each spatial entity consists of. Thus, when new data arrives in deployment, both these configuration processes will need to be performed again when using machine learning. In contrast, *SkyEx-T* is *robust for deployment*, while the machine learning techniques will require continuous configuration and re-tuning, making them fragile in a real deployment.
Summary. *SkyEx-T* outperforms machine learning on very small training sets and performs similarly for larger training sets. Furthermore, *SkyEx-T* is fully explainable, in contrast to the machine learning techniques, which behave like black boxes, with no possibility to comprehend their model. What is more, *SkyEx-T* is robust for deployment, while with machine learning, we might need to play with the parameters and re-configure them over and over as new data arrives.

5.3 SkyEx-T cut-off prediction

In this section, we evaluate the prediction of the cut-off c_t for *SkyEx-T*. The cut-off c_t is learned from the training set as in Algorithm D.1. Alternatively, we found the optimal cut-off c^* that would yield the highest F-measure for the learned preference function. To do so, we exhaustively tried all the cut-offs in the test set, measured the F-measure in each of them, and noted the highest value. In practice, this is not only time-consuming but even impossible since the labels are not present in the test set. By having the optimal cut-off, we can evaluate how accurately *SkyEx-T* can predict the cut-off. We repeated this procedure 10 times on disjoint training sets for each of the training sizes and report the averages in Table D.5 and Table D.6.
For both datasets, *SkyEx-T* finds a cut-off that is nearly-optimal. For small training sets (0.05% and 0.1% in North-DK and 4% in Restaurants), *SkyEx-T* yields an F-measure that is around only 0.025 smaller than the optimal, cor-

Training size	0.05%	0.1%	0.4%	0.8%	1%	4%	8%	12%	16%	20%	80%
SkyEx-T F-meas.	0.682	0.690	0.708	0.705	0.706	0.736	0.717	0.718	0.711	0.711	0.727
SkyEx-T F-meas. in c*	0.707	0.715	0.714	0.718	0.713	0.740	0.721	0.719	0.712	0.712	0.731
Diff F-meas.	0.025	0.025	0.006	0.014	0.007	0.004	0.004	**0.001**	**0.001**	**0.001**	0.005
Diff F-meas. in %	3.49%	3.46%	0.81%	1.90%	0.95%	0.48%	0.62%	**0.13%**	0.17%	**0.13%**	0.65%

Table D.5: SkyEx-T F-measure vs SkyEx-T F-measure in the optimal cut-off (c*) in North-DK

Training size	1%	4%	8%	12%	16%	20%	80%
SkyEx-T F-meas.	0.782	0.813	0.831	0.823	0.821	0.828	0.820
SkyEx-T F-meas. in c*	0.841	0.840	0.840	0.839	0.834	0.839	0.838
Diff F-meas.	0.059	0.027	**0.009**	0.015	0.013	0.011	0.018
Diff F-meas. in %	6.99%	3.26%	**1.07%**	1.84%	1.54%	1.31%	2.18%

Table D.6: SkyEx-T F-measure vs SkyEx-T F-measure in the optimal cut-off (c*) in Restaurants

responding to 3.4% difference. When using 1% of the data as a training set in Restaurants, *SkyEx-T* predicts a cut-off that results in an F-measure that is 0.059 smaller than the optimal. However, considering that 1% of the training set in the Restaurants dataset might contain only 1 or 2 positive pairs, the *SkyEx-T* prediction is highly satisfactory. The F-measure values improve rapidly when moving to larger training sets; here, the difference between the F-measures is only 0.01, a small 0.95% difference on average, for 0.04%-8% training sets on North-DK and on average 1.44% for 8%-20% training sets on Restaurants. Furthermore, for 12%-20% of the data as training set, the loss in F-measure in the North-DK dataset is 0.001 (0.13%), which is insignificant. *SkyEx-T predicts a cut-off that is near-optimal, with a difference of only 0.01 in F-measure (1.34%) in North-DK and 0.02 (2.6%) in Restaurants*. Thus, we successfully experimentally validated our theoretical findings of Theorem D.1, Theorem D.2, and Lemma D.1.

6 Conclusions

We addressed the problem of spatial entity linkage with a skyline-based approach, *SkyEx-T*, which, in contrast to the previous works [10, 11], trains the preference function and learns the cut-off for separating the classes, even in a tiny training set. We used better similarity features, computed with *LGM-X*, which specifically capture the characteristics of spatial entities. We showed that *SkyEx-T* outperforms the machine learning when using very small training sets and achieves similar results to them when increasing the training size, while offering a human-readable and easily explainable model. Moreover, in contrast to the machine learning techniques, *SkyEx-T* is robust and does not require further tuning. As future work, we plan to improve the scal-

ability of *SkyEx-T* and adapt it to other classification problems. Moreover, we will experiment with further tuning all the parameters of *LGM-X* and also the hyperparameters related to machine learning classifiers in order to achieve better results.

References

[1] J. Efremova, B. Ranjbar-Sahraei, H. Rahmani, F. A. Oliehoek, T. Calders, K. Tuyls, and G. Weiss, "Multi-source entity resolution for genealogical data," in *Population reconstruction*. Springer, 2015, pp. 129–154.

[2] D. Firmani, B. Saha, and D. Srivastava, "Online entity resolution using an oracle," *Proceedings of the VLDB Endowment*, vol. 9, no. 5, pp. 384–395, 2016.

[3] Q. Wang, D. Vatsalan, and P. Christen, "Efficient interactive training selection for large-scale entity resolution," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2015, pp. 562–573.

[4] M. Edwards, S. Wattam, P. Rayson, and A. Rashid, "Sampling labelled profile data for identity resolution," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 540–547.

[5] O. Goga, H. Lei, S. H. K. Parthasarathi, G. Friedland, R. Sommer, and R. Teixeira, "Exploiting innocuous activity for correlating users across sites," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 447–458.

[6] S. Isaj, N. B. Seghouani, and G. Quercini, "Profile reconciliation through dynamic activities across social networks," in *International Conference on Advanced Information Systems Engineering*. Springer, 2019, pp. 126–141.

[7] R. Karam, F. Favetta, R. Kilany, and R. Laurini, "Integration of similar location based services proposed by several providers," in *International Conference on Networked Digital Technologies*. Springer, 2010, pp. 136–144.

[8] B. Berjawi, E. Chesneau, F. Duchateau, F. Favetta, C. Cunty, M. Miquel, and R. Laurini, "Representing uncertainty in visual integration." in *DMS*, 2014, pp. 365–371.

[9] A. Morana, T. Morel, B. Berjawi, and F. Duchateau, "Geobench: a geospatial integration tool for building a spatial entity matching benchmark," in *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2014, pp. 533–536.

[10] S. Isaj, T. B. Pedersen, and E. Zimányi, "Multi-source spatial entity linkage," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[11] S. Isaj, E. Zimányi, and T. B. Pedersen, "Multi-source spatial entity linkage," in *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*, 2019, pp. 1–10.

[12] G. Giannopoulos and M. Meimaris, "Learning domain driven and semantically enriched embeddings for poi classification," in *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*, 2019, pp. 214–217.

[13] G. Giannopoulos, V. Kaffes, and G. Kostoulas, "Learning advanced similarities and training features for toponym interlinking," in *European Conference on Information Retrieval*. Springer, 2020, pp. 111–125.

[14] V. Kaffes, G. Giannopoulos, N. Karagiannakis, and N. Tsakonas, "Learning domain specific models for toponym interlinking," in *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2019, pp. 504–507.

[15] J. Lee, R. Hussain, V. Rivera, and D. Isroilov, "Second-level degree-based entity resolution in online social networks," *Social Network Analysis and Mining*, vol. 8, no. 1, p. 19, 2018.

[16] K. Shu, S. Wang, J. Tang, R. Zafarani, and H. Liu, "User identity linkage across online social networks: A review," *Acm Sigkdd Explorations Newsletter*, vol. 18, no. 2, pp. 5–17, 2017.

[17] P. Christen, T. Churches, and M. Hegland, "Febrl–a parallel open source data linkage system," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2004, pp. 638–647.

[18] G. Quercini, N. Bennacer, M. Ghufran, and C. N. Jipmo, "LIAISON: reconciLIAtion of Individuals Profiles Across SOcial Networks," in *Advances in Knowledge Discovery and Management*. Springer, 2017, pp. 229–253.

[19] L. Liu, W. K. Cheung, X. Li, and L. Liao, "Aligning users across social networks using network embedding." in *IJCAI*, 2016, pp. 1774–1780.

[20] Y. Zhang, J. Tang, Z. Yang, J. Pei, and P. S. Yu, "Cosnet: Connecting heterogeneous social networks with local and global consistency," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1485–1494.

[21] F. Li, M. L. Lee, W. Hsu, and W.-C. Tan, "Linking temporal records for profiling entities," in *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, 2015, pp. 593–605.

[22] G. Papadakis, E. Ioannou, C. Niederée, T. Palpanas, and W. Nejdl, "Beyond 100 million entities: large-scale blocking-based resolution for heterogeneous data," in *Proceedings of the fifth ACM international conference on Web search and data mining*, 2012, pp. 53–62.

[23] G. Papadakis, J. Svirsky, A. Gal, and T. Palpanas, "Comparative analysis of approximate blocking techniques for entity resolution," *Proceedings of the VLDB Endowment*, vol. 9, no. 9, pp. 684–695, 2016.

[24] L. Shu, A. Chen, M. Xiong, and W. Meng, "Efficient spectral neighborhood blocking for entity resolution," in *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 2011, pp. 1067–1078.

[25] V. Efthymiou, K. Stefanidis, and V. Christophides, "Big data entity resolution," in *2015 IEEE International Conference on Big Data (IEEE BigData 2015)*, 2015.

[26] R. Abdalla, "Geospatial data integration," in *Introduction to Geospatial Information and Communication Technology (GeoICT)*. Springer, 2016, pp. 105–124.

[27] P. Tabarro, J. Pouliot, R. Fortier, and L.-M. Losier, "A webgis to support gpr 3d data acquisition: A first step for the integration of underground utility networks in 3d city models," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, p. 43, 2017.

[28] S. Balley, C. Parent, and S. Spaccapietra, "Modelling geographic data with multiple representations," *International Journal of Geographical Information Science*, vol. 18, no. 4, pp. 327–352, 2004.

[29] M. Schäfers and U. W. Lipeck, "Simmatching: adaptable road network matching for efficient and scalable spatial data integration," in *Proceedings of the 1st ACM SIGSPATIAL PhD Workshop*, 2014, pp. 1–5.

[30] E. Moreau, F. Yvon, and O. Cappé, "Robust similarity measures for named entities matching," in *COLING 2008, 22nd International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, UK*, 2008, pp. 593–600.

[31] R. Santos, P. Murrieta-Flores, and B. Martins, "Learning to combine multiple string similarity metrics for effective toponym matching," *International Journal of Digital Earth*, vol. 11, 2018. [Online]. Available: https://doi.org/10.1080/17538947.2017.1371253

[32] C. Davis Jr and E. Salles, "Approximate string matching for geographic names and personal names," 01 2007, pp. 49–60.

[33] D. Kılınç, "An accurate toponym-matching measure based on approximate string matching," *Journal of Information Science*, vol. 42, pp. 138–149, 2016. [Online]. Available: https://doi.org/10.1177/0165551515590097

[34] G. Recchia and M. Louwerse, "A comparison of string similarity measures for toponym matching," *COMP 2013 - ACM SIGSPATIAL International Workshop on Computational Models of Place*, pp. 54–61, 11 2013.

[35] P. Christen, "A comparison of personal name matching: Techniques and practical issues," in *Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06)*, 2006, pp. 290–294.

[36] N. Dalvi, M. Olteanu, M. Raghavan, and P. Bohannon, "Deduplicating a places database," in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 409–418. [Online]. Available: https://doi.org/10.1145/2566486.2568034

[37] R. Santos, P. Murrieta-Flores, P. Calado, and B. Martins, "Toponym matching through deep neural networks," *International Journal of Geographical Information Science*, vol. 32, pp. 324–348, 2018. [Online]. Available: https://doi.org/10.1080/13658816.2017.1390119

[38] V. Sehgal, L. Getoor, and P. D. Viechnicki, "Entity resolution in geospatial data integration," in *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, 2006, pp. 83–90.

[39] A.-M. O. Raimond and S. Mustière, "Data matching–a matter of belief," in *Headway in spatial data handling*. Springer, 2008, pp. 501–519.

[40] S. Isaj and T. B. Pedersen, "skyex: an r package for entity linkage," in *International Conference on Extending Database Technology*, 2020, pp. 587–590.

[41] T. M. Cover and J. A. Thomas, "Information theory and statistics," *Elements of Information Theory*, vol. 1, no. 1, pp. 279–335, 1991.

[42] Y. Censor, "Pareto optimality in multiobjective problems," *Applied Mathematics and Optimization*, vol. 4, no. 1, pp. 41–59, 1977.

[43] M. Rosenblatt, "A central limit theorem and a strong mixing condition," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 42, no. 1, p. 43, 1956.

[44] W. H. Chiu, "Skewness preference, risk taking and expected utility maximisation," *The Geneva Risk and Insurance Review*, vol. 35, no. 2, pp. 108–129, 2010.

[45] S. Isaj and T. B. Pedersen, "Seed-driven geo-social data extraction," in *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*, 2019, pp. 11–20.

[46] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[47] W. A. Belson, "Matching and prediction on the principle of biological classification," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 8, no. 2, pp. 65–75, 1959.

[48] L. Breiman, "Random forests," vol. 45, no. 1, 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[49] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[50] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[51] R. Collobert and S. Bengio, "Links between perceptrons, mlps and svms," in *Proceedings of the Twenty-First International Conference on Machine Learning*, ser. ICML '04. Association for Computing Machinery, 2004, p. 23. [Online]. Available: https://doi.org/10.1145/1015330.1015415

[52] M. E. Kaminski, "The right to explanation, explained," *Berkeley Tech. LJ*, vol. 34, p. 189, 2019.

[53] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Adv. Large Margin Classif.*, 2000.

[54] C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn, "Bias in random forest variable importance measures: Illustrations, sources and a solution." bmc bioinformatics, 8(1), 25," *BMC bioinformatics*, vol. 8, 2007.

Paper E

Multi-Source Spatial Entity Linkage

Suela Isaj, Esteban Zimányi and Torben Bach Pedersen

The paper has been published in the
16th International Symposium on Spatial and Temporal Databases (SSTD'19)
August 19–21, 2019, Vienna, Austria, DOI:10.1145/3340964.3340979

Abstract

Besides the traditional cartographic data sources, spatial information can also be derived from location-based sources*. Location-based sources offer rich spatial information describing the semantics of locations. However, even though different location-based sources refer to the same physical world, each one has only partial coverage of the spatial entities of interest, describe them with different attributes, and sometimes provide contradicting information. Hence, the problem of finding which pairs of spatial entities belong to the same physical spatial entity demands specific attention. We propose a solution (*QuadSky*) to the problem of spatial entity linkage across diverse location-based sources.* QuadSky *starts with a spatial blocking technique (QuadFlex) that inherits the concept and the complexity from the quadtree algorithm but improves the splitting technique not to separate nearby points. After comparing the spatial entities of the same block, we propose a novel algorithm, referred to as* SkyEx *that separates the pairs considered as a match (positive class) from the rest (negative class) by using Pareto optimality.* SkyEx *does not require weights on the attributes, scoring function, or a training set.* QuadSky *achieves 0.85 precision and 0.85 recall for a manually labeled dataset of 1,500 pairs and 0.87 precision and 0.6 recall for a semi-manually labeled dataset of 777,452 pairs. Moreover,* QuadSky *provides the best trade-off between precision and recall and consequently, the best F-measure compared to the existing baselines.*

DOI:10.1145/3340964.3340979

1 Introduction

Web data and social networks are growing in terms of information volume and heterogeneity. Almost all online sources offer the possibility to introduce locations (geo-tagged entities accompanied by semantic details). A specific type of sources is *location-based sources*, whose primary focus is locations. In contrast to cartographic data sources, locations in location-based sources have a hybrid form that stands between a spatial object and an entity. We refer to them as *spatial entities* since they are spatially located but also identified by several other attributes such as the name of the location, the address, the phone, keywords, etc. Given that spatial entities provide richer semantics, several systems that rely on spatial information such as geo-recommender systems, selecting influential locations, search engines that use geo-preferences, etc. could improve further their results by using spatial entities instead of spatial objects. Hence, it is of interest to collect and integrate spatial entities from location-based sources.

While a spatial object is identified only by the coordinates, this is not the case for spatial entities. Different spatial entities might co-exist in the same coordinates (shops in a shopping mall), or the same entity might be located in different but nearby coordinates across different sources. The identity of a spatial entity is the combination of several attributes. Nevertheless, the identity of a spatial entity is sometimes difficult to infer due to the inconsistencies in the sources; each location-based source contains different attributes; some attributes might be missing and even contradicting. For example, the spatial entity "Lygten" is located in (57.436 10.534) and associated to keywords such as "coffee", "tea", and "cocoa and spices" in source A. In the meantime, source B contains the spatial entity named "Restaurant Lygten" in (57.435 10.533), described by the keyword "restaurant". We need a technique that can automatically decide whether these two spatial entities are the same real-world entity. The problem of finding which spatial entities belong to the same physical entity is referred to as *spatial entity linkage* or *spatial entity resolution*. According to [1], entity linkage establishes a link between spatial entities, whereas entity resolution goes further by merging related entities into one merged representative entity. Since we do not perform the latter, we use the term entity linkage.

There are several works that apply entity linkage in various fields [2–4, 4, 5, 5–10] but little regarding spatial entities [11–13]. In general, the entities studied in the current entity linkage paper refer to people; thus, the methodologies and the models are based on the similarities that two records of the same individual would reveal. Moreover, these works discard the spatial character of spatial entities. As for the works in spatial entity integration [11–13], their main contribution is a tool rather than an algorithm. What is more, the

methods propose arbitrarily attribute weights and score functions without experimentation nor evaluation. To sum up, *with the growing amount of information from location-based sources and the necessity for richer spatial information, a method to link spatial entities across different sources is needed*.
In this paper, we address the problem of spatial entity linkage across different location-based sources. First, we propose a method that uses the geocoordinates to arrange the spatial entities into blocks. Then, we pairwise compare the attributes of the spatial entities. Finally, we introduce a novel technique for deciding whether the pairs of compared entities belong to the same physical entity. Our contributions are:

1. We introduce *QuadSky*, a technique for linking spatial entities and we evaluate it on real-world data from four location-based sources.

2. We propose an algorithm called *QuadFlex* that organizes the spatial entities into blocks based on their spatial proximity. *QuadFlex* inherits the concept and the complexity of quadtrees but avoids assigning nearby points in different blocks.

3. To decide whether a of pair spatial entities refers to the same physical entity or not, we propose a flexible technique (*SkyEx*) that is based on the concept of Pareto optimality and needs no weights, no scoring functions, nor a training set.

The remainder of the paper is structured as follows: first, we describe and compare to state of the art in Section 2; then, we detail our proposed algorithm in Section 3; later, we provide experiments regarding the performance of our proposed solution in Section 4 and finally, we conclude in Section 5.

2 Related Work

The spatial entity linkage problem has not been explicitly addressed, but there are several works on *entity resolution* and also on *spatial data integration*. The general data integration problem covers three aspects: *schema matching*, *entity resolution* and *data fusion* [4, 14]. In general, there are various works that tackle the entity resolution problem but few that deal with spatial entities.

Entity resolution The *entity resolution* problem has been referred in the literature with multiple terms including *deduplication*, *entity linkage*, and *entity matching*. Entity resolution has been used in various fields such as matching profiles in social networks [2], bioinformatics data [3], biomedical data [15], publication data [4, 5], genealogical data [6], product data [4, 5], etc. The attributes of the entities are compared, and a similarity value is assigned. The

decision of whether to link two entities or not is usually based on a scoring function. However, finding an appropriate similarity function that combines the similarities of attributes and decides on whether to link or not the entities is often difficult. Several works use a training set to learn a classifier [7, 8, 16], others base the decision on a threshold derived through experiments [9, 17]. Other approaches that deal with uncertainty are described in the survey of Magnani and Montesi [18], including probabilistic, rule-based probabilistic, fuzzy, and preference-based relationships between records as well as aggregation of multi-matches. Finally, the decision on whether to link to entities or not can also be based on the feedback of an oracle [4, 5] or of a user [5].

Spatial data integration There are several works on integrating geographical data; some integrate purely spatial objects, some spatial entities. Spatial objects differ from spatial entities mainly because a spatial object is fully determined by the coordinates and the spatial shape whereas a spatial entity, in addition to being geo-located, has a well-defined identity (name, phone, opening hours, categories). The works on spatial object integration aim to create a unified spatial representation of the spatial objects that come from single/multiple sources. The solutions in [19–22] are purely spatial and discuss the integration of spatial objects originating from sensors and radars to have a better representation of the surface in 2D or even in 3D. Road network integration is tackled by Schafers at al [23] where rules are used to detect matching and non-matching roads. The matching is performed on the similarity of the roads in terms of the length, angles, shape, as well as the name of the street if available. Nonetheless, this approach is based on roads only and cannot apply to the linking of spatial entities.
Entity resolution of spatial entities has been discussed in [11–13, 24]. The work in [24] is a bridge between the works in spatial data integration and spatial entity linkage because the entities have names, coordinates, and types but similarly to spatial objects, they refer to landscapes (rivers, deserts, mountains, etc.). The method used in [24] is supervised and requires labeled data. Moreover, even the similarity of the attribute "type" is learned through a training set. Regarding [11–13], the main contribution of these works relies on designing a spatial entity matching tool rather than an integration algorithm. The authors of [13] provide a general matching technique for spatial entity matching. Spatial entities within a radius are compared with each other, and the value of the radius is fixed depending on the type of spatial entity. For example, the radius is 50 m for restaurants and hotels, but 500 m for parks. All attributes (except coordinates) are compared using the Levenshtein similarity. Since the name, the geodata and the type of the entity are always present, they carry two-thirds of the weight in the scoring function whereas the weights of the website, the address and the phone number are

tuned to one-third. The prototype of the spatial entity matching in [12] relies on a technique that arbitrarily uses an average of the similarity scores of all textual attributes without providing a discussing on this choice. Similarly to [11, 12], the main contribution of the work in [13] is designing a tool for spatial entity integration. The underlying algorithm considers spatial entities that are 5 m apart from each other and compares the name of the entities syntactically and the metadata related to an entity semantically. Finally, the decision is taken using the belief theory [25]. The works in [11–13] lack an evaluation of the algorithms.

Summary On the one hand, the general entity resolution approaches propose interesting solutions, but they do not consider the spatial character of a spatial entity. They do not deal with geographic coordinates and are designed to match entities that represent individuals (profiles in social networks, authors and publications, medical records, genealogical connections, etc.) or even linking species in nature. The proposed solutions for the former [entity resolution in individuals], either supervised or based on an experimental threshold, are learned on human entity datasets. One can not merely assume the resemblance of behaviors in a human entity dataset to a spatial entity one. The solutions in the latter [species in nature] are based on domain-specific algorithms that have little to no applicability in other fields. On the other hand, there is little specific work in spatial entities [11–13], mostly focusing on a tool for spatial data integration rather than on the algorithm. In all these works, the scoring function is chosen arbitrarily and no evaluation provided.

3 Spatial Entity Linkage

In this section, we introduce definitions, the problem of spatial entity linkage, and our proposed solution.

3.1 Problem Definition

The basic concept used in this work is a *spatial entity*, used for locations, places, businesses, etc. Spatial entities originate from location-based sources such as directories with location information (yellow pages, Google Places, etc.) and location-based social networks (Foursquare, Gowalla, etc.).

Definition E.1. *A spatial entity s is an entity identified uniquely within a source I, located in a geographical point p and accompanied by a set of attributes $A = \{a_i\}$.*

The attributes connected to an s can be categorized as:

- *Spatial*: the point where the entity is located, expressed in longitude and latitude.

- *Textual*: attributes that are in the form of text such as name, address, website, description, etc.
- *Semantic*: attributes in the form of text that enrich the semantics behind a spatial entity. Examples of this type are categories, keywords, metadata, etc.
- *Date, time or number*: other details about a spatial entity such as phone, opening hours, date of foundation, etc.

An example of a spatial entity originating from Yelp can be a place named "Star Pizza" in the point (56.716 10.114), with the keywords "pizza, fast food", and with address "Storegade 31". The same spatial entity can be found again in Yelp or other sources, sometimes having the same attributes, more, less or even attributes with contradictory values. Thus, there is a need for an approach that can unify information within and across different sources in an intelligent manner.
Problem definition: Given a set of spatial entities S originating from multiple sources, the spatial entity linkage problem aims to find those pairs of spatial entities $\langle s_i, s_j \rangle$ *that refer to the same physical spatial entity.*
In the following section, we introduce *QuadSky*, our solution to the spatial entity linkage problem.

3.2 QuadSky Approach

We propose *QuadSky*, a solution based on a quadtree data partitioning and skyline exploration. The overall approach is detailed in Fig. E.1. *QuadSky* consists of three main parts: spatial blocking (*QuadFlex*), pairwise comparisons and labelling the pairs (*SkyEx*). *S* contains all spatial entities from all sources *I*. We propose *QuadFlex*, a quadtree-based solution that can perform the spatial blocking by respecting the distance between spatial entities and the density of the area. The output of *QuadFlex* is a list of leaves with spatial entities located nearby. Within the leaves, we perform the pairwise comparisons of the attributes. After this second phase, we obtain a list of pairs of spatial entities and their similarities of attributes. In order to decide which pairs dictate a match and which not, we propose a novel technique, referred to as *SkyEx* that explores *k* skylines (concepts detailed in Section 3.5) of the pairs. *SkyEx* separates the pairs that refer to the same physical spatial entity (the positives class) from the rest (the negative class). In the following sections, we detail each of the phases of *QuadSky*.

3.3 Spatial Blocking

Performing all possible comparisons between pairs of spatial entities is time-consuming. Since spatial proximity is a strong indicator in finding a match,

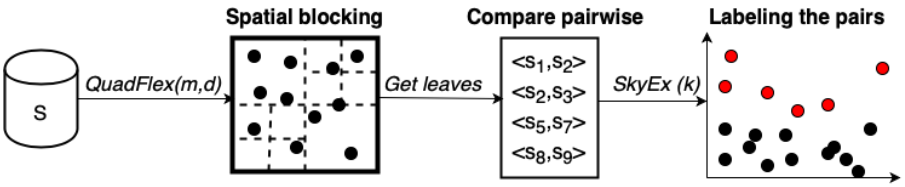


Fig. E.1: QuadSky approach

the first step is to group nearby spatial entities in blocks. Several generic blocking techniques have been discussed in [26, 27], including methods based only on one attribute or several ones, block purging (discards the blocks that are over the size limit), etc. However, these techniques are mostly based on textual attributes and not applicable to spatial blocking. We propose a quadtree-based solution (QuadFlex) that uses a tree data structure but also preserves the spatial proximity of spatial entities. Our contribution is twofold; we use quadtrees (meant for spatial indexing) for spatial blocking, and we modify the recursive procedure of the quadtrees to accommodate more points that are nearby instead of splitting them arbitrarily in different children.

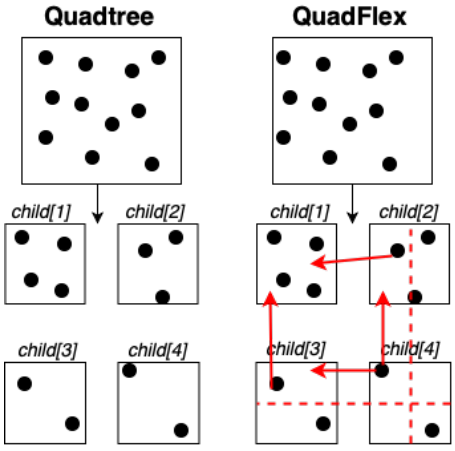


Fig. E.2: QuadFlex versus quadtree

A quadtree is a tree whose nodes are always recursively split into four children when the capacity is filled [28]. Quadtrees of n points and d depth can be constructed in $O((d+1)n)$ time. After the quadtree is constructed, the points that fall in the same leaf are nearby spatially. Hence, these leaves are good candidates to be spatial blocks. There are several issues with the existing quadtree algorithm. First, a quadtree needs a capacity (number of

points) as a parameter. The capacity is not a meaningful parameter for spatial blocking, while the density of the area is a better candidate. For example, if the area is too dense (e.g., city center), even though the capacity is not reached, a further split would be more beneficial. Spatial entities in the city center tend to be nearby, but they are more unlikely to be the same. On the contrary, two points in the countryside (e.g., a farm) might be farther apart, but they still might be the same entity. Second, a quadtree does not limit the distance between points. Even though two points might be in an area that respects the density, if they are quite distant from each other, it is not necessary to compare them. The maximal distance between two points in a child is the diagonal of the area (all quadtree children are rectangular). We used m, the diagonal of an area, as a parameter that controls the distance of points rather than comparing all distances. Finally, a quadtree splits into four children, and sometimes nearby points might fall into different leaves. Hence, we might miss good pairs. We modify the procedure of the assignment of the points into a child by allowing more than one assignment.

Fig. E.2 shows the modifications that we do to the construction of the traditional quadtree for our version *QuadFlex*. The traditional quadtree divides the area of each parent into four smaller areas, the children. A point belongs only to one child. In our modification, the area will split similarly to the quadtree, but when we assign a point to a child, we will consider including points that fall near the border also in the current child. For example, in Fig. E.2, the red dashed line shows the area that will be considered to include points in the neighbors. We will add two points in child[0], one coming from child[1] and one from child[3]. Similarly, child[1] and child[3] will receive a point from child[4].

Algorithm E.1 details the procedure for retrieving the spatial blocks with *QuadFlex*. The algorithm creates the root of the *QuadFlex* tree with the bounding box of the data and parameters m and d (line 1). Then, it inserts each spatial entity into the QuadFlex (line 4) and finally returns its leaves. The methods $insert(s)$ and $getIndex(s)$ are self calls on the *QuadFlex* object (*this*). The insertion procedure is similar to the traditional quadtree except that the constraint is not the capacity but the diagonal of the area m (maximal distance between points) and the density of the area d. Hence, if the diagonal of the QuadFlex is more than the distance m or the density is larger than our defined value d (line 12), the QuadFlex, similarly to a quadtree, will split into four children. However, in contrast to the traditional quadtree, a spatial entity might belong to more than one child. The method $getIndex(s)$ gets the list of indexes of the children where the new point will be assigned. Even though Q splits into 4 children, the lines *vertical* and *horizontal* (corresponding to the red dashed lines in Fig. E.1) allow a logical overlap of the areas and thus, neighboring spatial entities will not be separated.

Algorithm E.1 QuadFlex algorithm

Input: A set of entities $S = \{s_i\}$, diagonal m, density d
Output: The leaves QuadFlex Q *Q.leaves()* ;

Create $Q(m, d)$ where Q has the dimensions of the bounding box of S
for each s in S **do**
    Q.insert(s) // *Insert s into the QuadFlex*
end for
return *Q.leaves()*

Method *insert (s)*
if *this.children* $\neq \varnothing$ **then**
    Indexes $\leftarrow$ *getIndex(s)* // *Find where s belongs*
    for each i in *Indexes* **do**
        this.child[i].insert(s) // *Insert s to the children it belongs*
    end for
end if
if *this.diagonal* $> m$ **or** *this.density* $> d$ **then**
    Split the current object *this* into 4 children // *The area is larger or denser than our restrictions, so split as in the traditional quadtree*
end if
Indexes $\leftarrow$ *getIndex(s)*
for each i in *Indexes* **do**
    this.child[i].insert(s)
end for
return

Method *getIndex (s)*
Let *vertical* be the line that passes at 0.75 of the width of *this*
Let *horizontal* be the line that passes at 0.75 of the height of *this*
if s is left of *vertical* and above *horizontal* **then**
    Indexes.add(1) // *s fits in child[1]*
end if
if s is right of *vertical* and above *horizontal* **then**
    Indexes.add(2) // *s fits in child[2]*
end if
if s is left of *vertical* and below *horizontal* **then**
    Indexes.add(3) // *s fits in child[3]*
end if
if s is right of *vertical* and below *horizontal* **then**
    Indexes.add(4) // *s fits in child[4]*
end if
return *Indexes*

3.4 Pairwise Comparisons

After the spatial blocking, we perform a pairwise comparison of spatial entities that fall in the same leaf. Next, we describe the metrics for different types of attributes.

Textual Similarity We measure the textual similarity of spatial entities using the edit distance between the words. The Levenshtein distance [29] between string s_1 and string s_2 $d(s_1, s_2)$ is the number of edits (insertion, deletion, change of characters) needed to convert string s_1 to string s_2. We define the similarity as:

$$TextSim(s_1, s_2) = (1 - \frac{d(s_1, s_2)}{max(|s_1|, |s_2|)}) \tag{E.1}$$

Example E.1
Let us consider "Skippers Grill" and "Skippers Grillbar". The Levenshtein distance to convert "Skippers Grill" to "Skippers Grillbar" is 3 (3 insertions). The lengths of the first and the second string are 14 and 17 respectively. So, $TextSim$("Skippers Grill", "Skippers Grillbar") $= 1 - (3/max(14, 17) = 0.8235$.

Note here that not all textual attributes can be handled similarly. String similarity metrics are usually appropriate for attributes like names, usernames, etc. Some other textual attributes require other metrics that need to be customized. In this paper, we consider address as a specific textual attribute. The similarity between two addresses cannot be measured with Levenshtein, Jaccard, Cosine, etc. since a small change in the address might be a giant gap in the spatial distance between the entities. For example, "Jyllandsgade 15 9480 Løkken" and "Jyllandsgade 75 9480 Løkken" have a distance of 1 and Levenshtein similarity of 0.963, but they are 650 meters apart. However, "Jyllandsgade 15 9480 Løkken" and "Jyllandsgade 15 9480 Løkken Denmark" have a distance of 8 and Levenshtein similarity of 0.772, but they are the same building. In [11, 12] the address is considered as another textual attribute without using its real semantics. In our case, we perform some data cleaning (removing commas, punctuation marks, lowercase, etc.), and then we search for equality ($s_1 = s_2$) or inclusion ($s_1 \subset s_2$ or $s_2 \subset s_1$). We assign a similarity of 1 in the case of equality and 0.9 in the case of inclusion. Otherwise, the strings are not considered the same.

Semantic Similarity The similarity of fields like categories, keywords or metadata cannot be compared only syntactically. Sometimes, several synonyms are used to express the same idea. Thus, we need to find a similarity

than considers the synonyms as well. We use Wordnet [30] for detecting the type of relationship between two words and Wu&Palmer similarity measure (*wup*) [31], which calculates how related two words are by taking into account the depths of the two synsets (sets of synonyms) in WordNet, along with the depth of the Least Common Subsumer (the most specific concept that is an ancestor of both words). The semantic similarity between two spatial entities is the maximal similarity between their list of categories,keywords or metadata. The semantic similarity of the spatial entities s_1 and s_2 is:

$$SemSim(s1, s2) = max\{wup(c_i, c_j)\} \tag{E.2}$$

where $c_i \in C_1$ and $c_j \in C_2$ and C_1 is the set of keywords of s_1 and C_2 is the set of keywords s_2.

Example E.2
Let us take an example of two spatial entities s_1 and s_2 and their corresponding semantic information expressed as keywords $C_1 = \{$"restaurant","italian"$\}$ and $C_1 = \{$"food","pizza"$\}$. The similarity between each pair is $wup($"restaurant","food"$) = 0.4$, $wup($"italian","food"$) = 0.4286$, $wup($"restaurant","pizza"$) = 0.3333$ and $wup($"italian","pizza"$) = 0.3529$. Finally, the semantic similarity of s_1 and s_2 is $SemSim(s1, s2) = max\{0.4, 0.4286, 0.3333, 0.3529\} = 0.4286$.

Date, time or numeric similarity The similarity between two fields expressed as numbers, dates, times or intervals is simply a boolean decision (true or false). For example, if the phone numbers change with only one digit, they are still different. Even though the similarity of these fields relies only on an equality check, most of the effort is put in data preparation. For example, before the comparison, the different phone formats should be identified and cleaned from prefixes. Other data formats like intervals (opening hours) might require temporal queries for similarity, inclusion, and intersection of the intervals. In this paper, we do not use these attributes for measuring the similarity between spatial entities, but for constructing the ground truth.

Summary Spatial entities are characterized by different spatial attributes that differ in formats but also in the semantics. Textual attributes are the most studied in the literature, and several similarity metrics have already been proposed. We use Levenshtein while comparing the names of spatial entities, and customized comparisons for the address. Thus, we check for equality or inclusion to detect similar addresses. Metadata, keywords, categories, etc. are attributes that carry semantics. Since textual similarity cannot

measure the similarity between synonyms, we rely on Wordnet to detect similar keywords attached to the spatial entities. Finally, other formats such as numbers, intervals, timestamps, etc. can be checked for equality, but also some background knowledge might be needed to detect matches.

3.5 Labeling the Pairs

After the pairwise comparison, the pairs are represented as points in a space with n dimensions, where each dimension is an attribute. A pair has n similarity values, one for each attribute. We denote as δ_a the similarity of two spatial entities for attribute a. For example, a pair $\langle s_1, s_2 \rangle$ is represented as $(\delta_{a_1}, \delta_{a_2} ... \delta_{a_n})$. The problem that we need to solve is which $\langle s_i, s_j \rangle$ pairs indicate a strong similarity and should be considered for a match. This is an old problem in the entity linkage community. A classifier [7, 8, 32] can learn the behavior of the matches and detect the positive class. However, it is difficult to obtain labeled data, especially across different sources. Moreover, there is always the risk of overfitting when training a classifier on a sample. Some works assign weights on the similarity scores and test different combinations [9, 17, 32]. The problem that arises with these methods is that finding the best combination might require extensive experiments and might overfit the data.

We propose a more relaxed technique that uses Pareto optimality [33] for filtering the positive class. *A solution (x, y) is Pareto optimal when no other solution can increase x without decreasing y*. The points in the same Pareto frontier or skyline have the same utility. Widely used in economics and multi-objective problems, Pareto optimality is free of weights and similarity score functions. In the context of entity resolution, the skylines provide a selection of points that are better than others, but without quantifying how much better. The pairs that refer to the same physical spatial entity (the positive class) are expected to have high values of δ. In general, the positive class is the minority and is spread all over the dataset, resembling outliers. This means that the first Pareto optimal frontier might contain only a couple of points. Thus, an exploration of several skylines (k levels) is needed. Under the assumption that the best values of δ belong to the pairs from the positive class, we label the k skylines as the positive class and the rest as the negative. *To the best of our knowledge, we are the first to propose a Pareto optimal solution for detecting matches for an entity linkage/resolution problem*.

Definition E.2. *An attribute a is positive discriminating if its similarity δ_a indicates a positive class rather than a negative.*

An example of a positive discriminating attribute is the similarity of name. A higher name similarity is more likely to indicate a match than a non-match. For example, the name similarity for *Mand & Bil* and *Mand og Bil* is 0.75, and

for *Solid* and *Sirculus ApS* is 0.16 . Hence, the former pair has a higher probability of being a match than the second. Examples of negative discriminating attributes are the edit distance between two names. If the distance between the names is high, then the pairs are less likely to be a match.

Definition E.3. *The utility of a positive discriminating attribute a to the positive class, denoted as u_a, is a monotonically increasing function that quantifies the contribution of the similarity of attribute δ_a to indicate a match.*

Each attribute contributes to the labeling problem. A higher similarity δ_a of a has a higher utility than a lower value of δ_a. Hence, if $\delta_a(\langle s_1, s_2\rangle) > \delta_a(\langle s_3, s_4\rangle)$, then $u_a(\langle s_1, s_2\rangle) > u_a(\langle s_3, s_4\rangle)$.

Definition E.4. *The utility of a pair denoted as $u(\langle s_i, s_j\rangle)$ is sum of the utilities of each of the attributes. $u(\langle s_i, s_j\rangle) = \sum_{i=1}^{n} u_{a_i}$.*

Note that the utility of a pair is not the sum of the similarities of the attributes ($u(\langle s_i, s_j\rangle) \neq \sum_{i=1}^{n} \delta_{a_i}$) but the sum of their utilities ($u(\langle s_i, s_j\rangle) = \sum_{i=1}^{n} u_{a_i}$). Nevertheless, $u(\langle s_i, s_j\rangle) = \sum_{i=1}^{n} \delta_{a_i} = \sum_{i=1}^{n} u_{a_i}$ is a specific case.

Definition E.5. *A skyline of level k denoted as $Skyline(k)$ is the collection of pairs $\langle s_i, s_j\rangle$ of equal utility such that $u_{Skyline(k)} > u_{Skyline(k-1)}$.*

Obviously, $Skyline(1)$ is the Pareto optimal frontier with the best values of δ_a. In order to continue with $Skyline(2)$, the points of $Skyline(1)$ are removed, and the frontier is calculated again. Every time we explore level k, the values in $Skyline(k)$ are the ones with the highest utility. This means that *there is no other point in a lower level that can bring a higher utility to the positive class*. This procedure continues for k steps. Algorithm E.2 formalizes our proposed procedure Skyline Explore (*SkyEx*) for labelling the pairs. The input is the set of pairs P produced from the *QuadFlex* blocking technique and the number of skyline levels k that we will explore. We find the points with the best combinations of δ that dominate the rest of the points and consequently, have a higher utility (line 3). Then, we put these points in P^+ and remove them from P (line 5). After all k levels are computed, we return the positive set of pairs P^+ and the negative P^-.
In contrast to techniques that use a similarity score function, *SkyEx* abstracts the concept of utility. Thus, no weights or similarity function is needed. Given that the points with a high utility are generally scattered, *SkyEx* can detect the positive class better than a clustering technique, which would fail in clustering together the positive class. Moreover, the flexibility of *SkyEx* makes it applicable to all problems where the expert knowledge on the contribution of the attributes is missing. Finally, *SkyEx* does not learn any behavior, so there is no risk of overfitting.

Algorithm E.2 Skyline Explore (SkyEx)

Input: A set of pairs $P = \{\langle s_i, s_j \rangle\}$, a number of skyline levels k
Output: A set of positive pairs P^+, a set of negative pairs P^- ;

1: $P^+ \leftarrow \varnothing$
2: **for** m in $[1, k]$ **do**
3: Filter $Skyline(m) = \{\langle s_i, s_j \rangle\} \mid \forall \langle s', s'' \rangle \in P - \{\langle s_i, s_j \rangle\}$, $u(\langle s_i, s_j \rangle) > u\langle s', s'' \rangle\}$ *// Find the Skyline*
4: Add $Skyline(m)$ to P^+ *// Label the skyline pairs as positive*
5: $P = P - Skyline(m)$
6: **end for**
7: $P^- \leftarrow P$ *// Label the rest as negative*
return P^+, P^-

4 Experiments

4.1 Dataset Description

The spatial entities that will be used in these experiments originate from four sources, namely Google Places (GP), Foursquare (FSQ), Yelp, and Krak. Krak (www.krak.dk) is a location-based source that offers information about companies, enterprises, etc. in Denmark and is also part of Eniro Danmark A / S., which publishes The Yellow Pages. The data is obtained by using the available APIs and the algorithm detailed in [34]. The distribution of the spatial entities can be observed in Fig. E.3. The dataset consists of 75,541 spatial entities where 51.50% comes from GP, 46.22% from Krak, 0.03% from FSQ, and 2.23% from Yelp. All the sources internally might contain possible links, so we need to compare entities within and outside the sources.

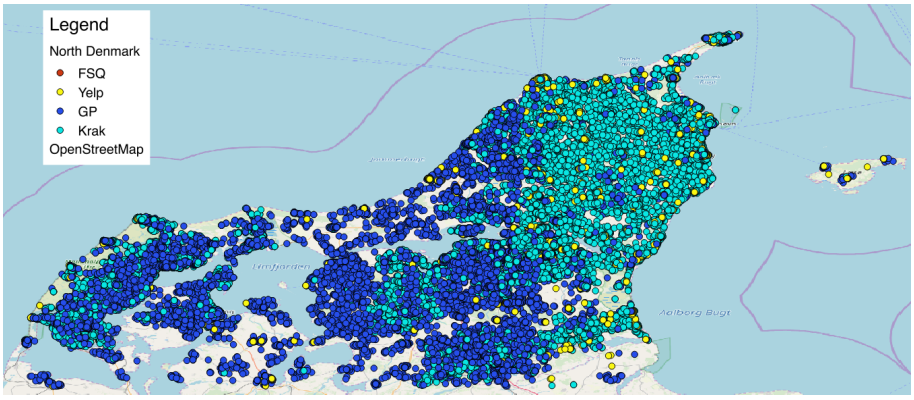
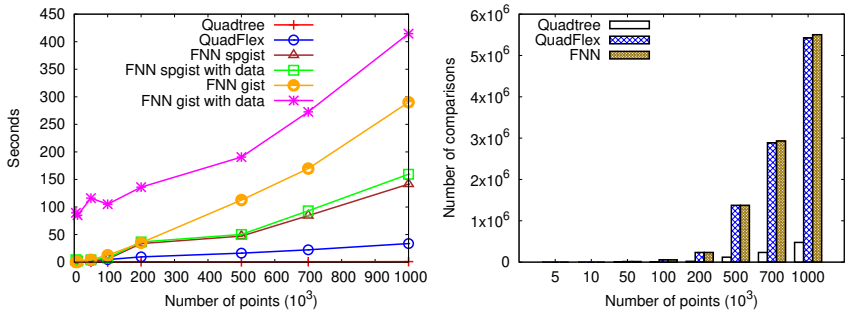


Fig. E.3: North Denmark dataset

4.2 QuadFlex Performance

In this section, we compare the performance of our proposed blocking technique to the traditional quadtree and Fixed Radius Nearest Neighbors algorithm [35] (FNN). FNN finds the neighbors that fall within a fixed radius from each point. *QuadFlex* and the quadtree algorithm are implemented in Java, while FNN is run on a Postgres database (https://www.postgresql.org) using spatial indexes; more specifically, two spatial indexes: GiST (https://www.postgresql.org/ docs/current/gist.html) (optimized C implementation of B-trees and R-trees) and SP-GiST (https://www.postgresql.org/docs/current/ spgist.html) (optimized C implementation of quadtrees and k-d trees). Our original dataset contains 75,541 entities in the whole North Denmark region (around 16 towns, 7,933 km^2), so the density is not high. A high data density means more neighbors and consequently, more pairs to compare. In order to test our *QuadFlex* on different data densities, we simulate up to 1,000,000 random points from Aalborg (139 km^2).

Fig. E.4 shows the comparison of quadtree, *QuadFlex* and FNN in terms of execution time (Fig. E.4a) and number of comparisons (Fig. E.4b). The FNN versions with data are computed on the database, and then the pairs are loaded back to the java implementation. The quadtree has the lowest execution time, followed by *QuadFlex*. FNN SP-GiST is comparable and sometimes even better than *QuadFlex* for small datasets. However, when the size of the dataset increases, *QuadFlex* manages to maintain an execution time that is eight times less than FNN GiST and 3 times less than FNN SP-GiST. FNN with SP-GiST index outperforms FNN GiST for all dataset sizes. As for the number of comparisons, *QuadFlex* enumerates 12 times more comparisons than quadtree, so our technique for not missing nearby pairs turned out effective. Moreover, *QuadFlex* contains almost all (99.99%) comparisons of



(a) Execution time

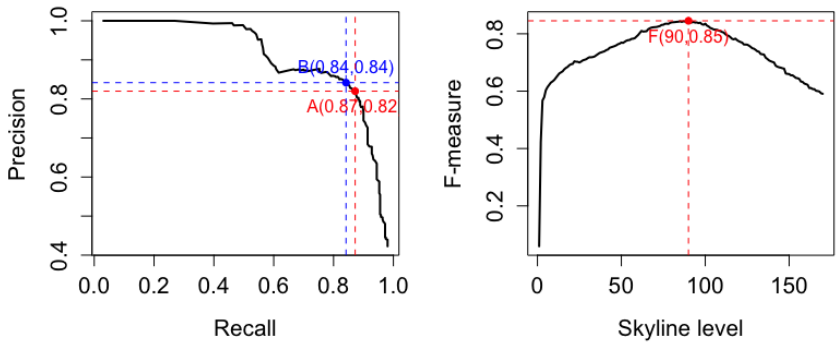
(b) Number of comparisons

Fig. E.4: Comparing quadtree, QuadFlex and FNN

FNN, compared to quadtree that contains only 10% of FNN. Furthermore, given that the scalability of *QuadFlex* is better than FNN, and *QuadFlex* is independent of the database implementations, the loss of around 0.01% of comparisons is insignificant. Moreover, this difference in comparisons can simply be explained by the fact that *QuadFlex* uses a rectangular area with diagonal of m meters, whereas FNN uses a circle with radius $\frac{m}{2}$. In the case of a square with diagonal m, the surface will be $\frac{m^2}{2}$, but for the circle with the diameter m the surface is $\pi\frac{m2}{4}$. So, the surface of the circle is $\frac{\pi}{2}$ times more.

4.3 SkyEx results

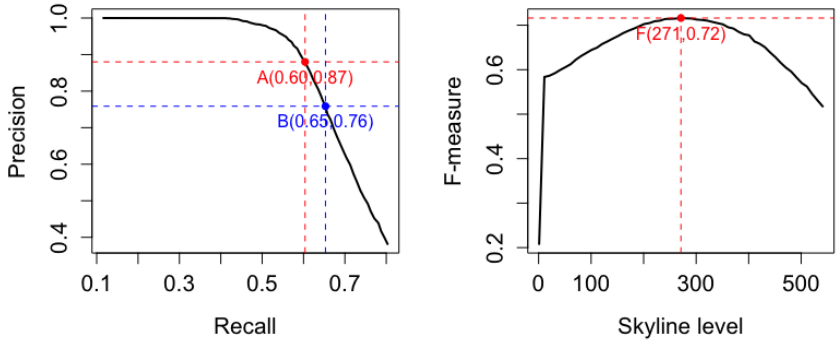
In this section, we evaluate the results of our proposed *SkyEx*. In the context of our problem, we define true positives TP as pairs that refer to the same physical entity and correctly labeled as positives, true negatives TN as pairs referring to different physical entities and correctly labeled as negatives , false positive FP as pairs that do not refer to the same physical entities but wrongly labeled as positives, and FN as pairs that refer to the same physical entity but wrongly labeled as negatives. We measure $precision = \frac{TP}{TP+FP}$, $recall = \frac{TP}{TP+FN}$ and *F-measure* $(F1) = 2\frac{precision*recall}{precision+recall}$.
We ran *QuadFlex* with 100 m and no density restriction, and we obtained 777,452 pairs. Having the same website or phone is a strong indicator of a match, so we use these attributes to infer the label. We refer to this labeling as *automatic labeling*. However, cases with different phone number or website but still the same entity, or same phone number but different entity might occur. For example, an entrepreneur who owns a fishing company and also a restaurant might use the same phone for both. Another example is the case of two different phones for the same entity on different sources. For this reason, we manually checked the labels of a sample of 1,500 pairs of entities, sometimes checking them even on maps and on the original sources. We will refer to the sample of manually checked pairs as D_{sample} and to the full dataset as D_{full}. Checking the labels manually on the full dataset of 777,452 pairs is unfeasible. Hence, we checked around 10,000 of the pairs, and for the rest, we rely on automatic labeling.
The results of *SkyEx* on D_{sample} and D_{full} are presented in Figs. E.5 and E.6. The curves in Figs. E.5a and E.6a shows the evolution of *precision* (y-axis) and *recall* (x-axis) while we move from one skyline to the next. As expected, *the more we explore, the more likely it is to retrieve more true positives and thus, improve the recall*. However, *the more we explore and label pairs as positives, the more likely it is to increase the number of false positives, so the precision degrades*. The algorithm explores several trade-offs, where points A and B are among the best. Point A with 0.87 *precision* and 0.82 *recall* in Fig. E.5a is the same best point in terms of *F-measure* as well. Fig. E.5b shows the levels of the



(a) Precision and recall **(b)** F-measure

Fig. E.5: SkyEx performance on D_{sample}

skyline, and the value of *F-measure* achieved. The highest value is 0.85 that corresponds to $k = 90$. Point B is also a good candidate with 0.84 *precision* and *recall*.

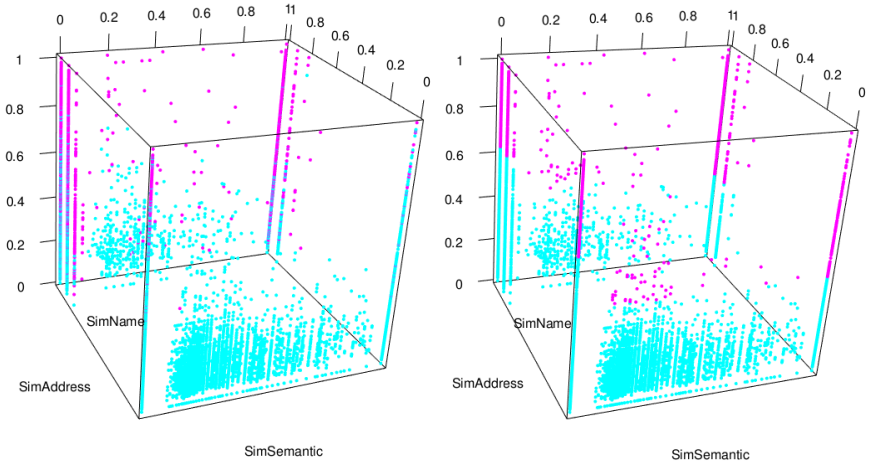


(a) Precision and recall **(b)** F-measure

Fig. E.6: SkyEx performance on D_{full}

The evaluation on the full dataset yields lower values compared to the sample, which might be a simple consequence of automatic labeling. Even though the labels are not all checked, *precision* and *recall* in D_{full} yield satisfactory values. Two of the best combinations are points A and B, where A is the combination also with the highest *F-measure* of 0.72 and 271 skyline

levels Fig. E.6b. *A* offers 0.6 *recall* and 0.87 *precision* while *B* offers a higher *recall* of 0.65 but a lower *precision* of 0.76. To have an idea of the real classes in D_{full} and the skyline, we plotted their distribution in Fig. E.7. Fig. E.7a shows the actual positive classes in pink and the negative ones in sky blue. It is noticeable that the positive class pairs are allocated in the highest values of the dimensions. *SkyEx* with 271 levels (Fig. E.7b) is able to capture this behaviour and achieve 0.6 *recall* and 0.87 *precision*. Despite the differences between both plots, *SkyEx* shows promising results in separating the positive class from the negative one.



(a) Actual classes **(b)** SkyEx classes

Fig. E.7: Positive (in pink) versus negative (in sky blue) classes for actual (a) and SkyEx (b) results

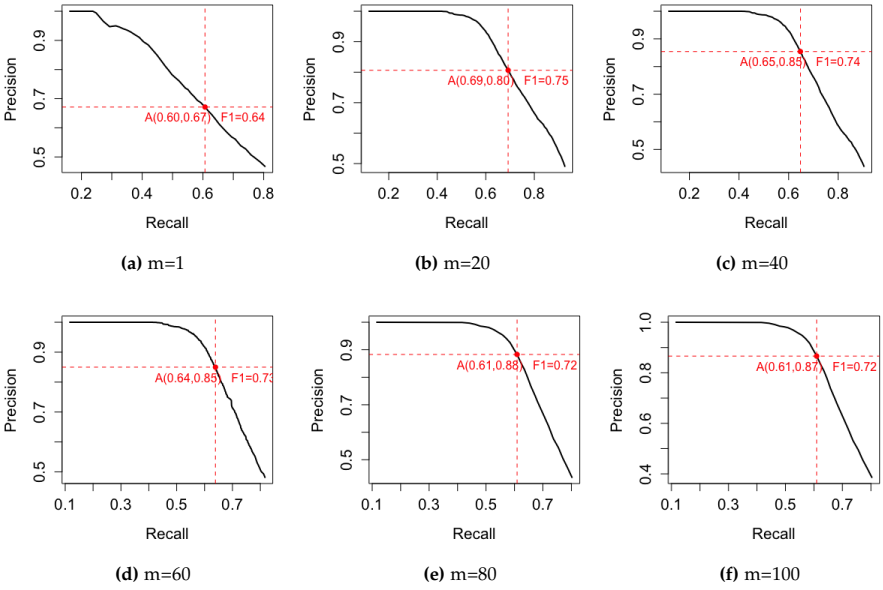
4.4 Experimenting with Different QuadFlex Parameters

So far, we used *QuadFlex* blocking technique with 100 meters and no density restriction. In this section, we will evaluate our approach *QuadSky* for different blocking parameters.

Meters	1	20	40	60	80	100
Total	41053	118437	226331	372553	557421	777452
% of pos	17.11%	19.88%	11.28%	7.06%	4.82%	3.49%

Table E.1: Dataset characteristics for different m

Changing m, no density limit In this experiment, we test different values of m used in *QuadFlex*for creating spatial blocks. We test m values of 1, 20, 40, 60, 80, and 100 meters. The size of the dataset for each of them is presented in Table E.1. The spatially close points are likely to be a match. Hence, the percentage of the positive class is generally higher for smaller values of m. An interesting case is $m = 1$, where the percentage of the positive class is lower than $m = 20$. One would expect that points that are 1 meter apart would unquestionably be a match. However, this is not always the case. Shopping malls, buildings that host several companies, etc. are characterized by the same coordinates but not necessarily the same spatial entities.



(a) m=1 **(b)** m=20 **(c)** m=40

(d) m=60 **(e)** m=80 **(f)** m=100

Fig. E.8: Performance of SkyEx for different m, no density limit

The results for different values of m are presented in Fig. E.8. The point A is the value with the highest *F-measure* ($F1$). For all cases, the *recall* is higher than 0.6. The *precision* is higher than 0.8 for all values of m, except $m = 1$, where the *precision* is 0.67. For $m = 1$, the positive and negative class are mixed, thus *SkyEx* loses a bit in *precision*. This is also an argument against the works that merge arbitrarily points that are 5 m apart. *Spatial proximity is not a definitive indicator of a match.*

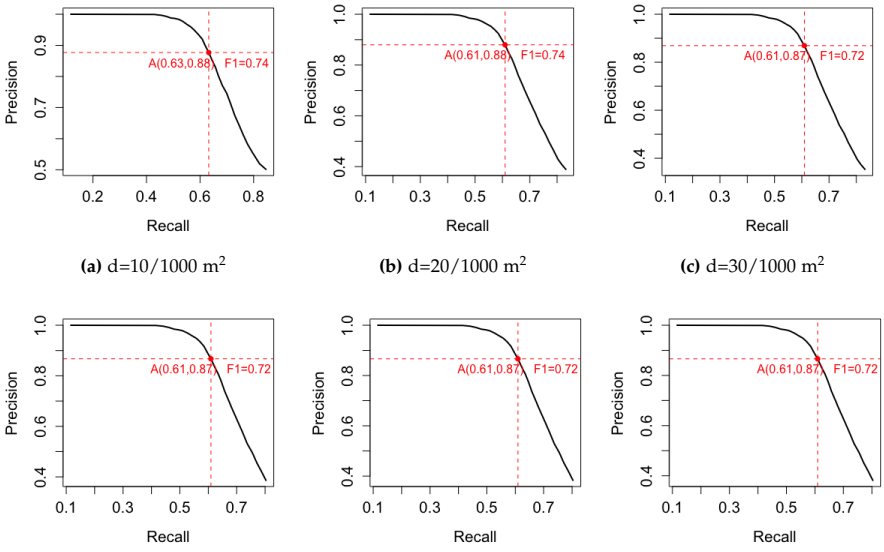
Changing d, m $\leq$ 100 The density is another parameter of *QuadFlex* that helps in creating smaller blocks in dense areas and larger ones in sparse areas. In this experiment, we test different values of density d and its effect

on the results. The size of the dataset and the percentage of the classes in Table E.2. When the density is smaller, we force *QuadFlex* to split further and create smaller blocks. Thus, the number of pairs reduces. Note that, on the contrary, the percentage of positives increases. Indeed, further splits allow us to create better blocks containing a higher percentage of positives. However, when the density increases above $\frac{30s}{1000m^2}$, fewer and fewer blocks are split further, so the dataset size and the percentage of the positives do not vary significantly.

Density	$\frac{10s}{1000m^2}$	$\frac{20s}{1000m^2}$	$\frac{30s}{1000m^2}$	$\frac{40s}{1000m^2}$	$\frac{50s}{1000m^2}$	$\frac{60s}{1000m^2}$
Total	290653	590583	711423	754195	770987	776664
% pos	8.61%	4.57%	3.81%	3.59%	3.51%	3.49%

Table E.2: Dataset characteristics for different d

The results after running *SkyEx* are presented in Fig. E.9. In all the cases, the *recall* stays above 0.61 and the *precision* above 0.87. A slightly better *precision* (0.88) and *recall* (0.63) is achieved in the case of a density of $\frac{10s}{1000m^2}$ (the lowest parameter). From both experiments with *QuadFlex* parameters, we can conclude that *SkyEx adapts very well in finding the correct classes even when the size of blocks change and even when the percentage of positives over negatives varies*.



(a) d=10/1000 m^2 **(b)** d=20/1000 m^2 **(c)** d=30/1000 m^2

(d) d=40/1000 m^2 **(e)** d=50/1000 m^2 **(f)** d=60/1000 m^2

Fig. E.9: Performance of SkyEx for different d, m=100

4.5 Comparison with Baselines

Even though there are several papers in spatial data integration, the works of [11–13] are the most similar to ours, as the rest of the related work considers only spatial objects, not spatial entities, or uses supervised learning techniques. We will compare *QuadSky* to the following baselines:

1. Berjawi et al. [12] propose Euclidean distance for the geographic coordinates and Levenshtein similarity for all other attributes. The similarities are integrated into a global similarity computed as a simple sum of the attribute similarities. The attributes mentioned in the paper are the name and the phone. However, since the phone is part of our automatic labeling, it can not be used in the algorithm as well. The authors consider pairs with $score \geq 0.75$ as a match with high confidence. We compare to this technique using name + address + geographic coordinates (V1) and name + geographic coordinates (V2).

2. Morana et al. [13] suggest filtering entities that share the same category or a token in the name. Then these entities are compared using the Euclidean distance for the coordinates, Levenshtein for the address and name, and Resnik similarity (Wordnet) for the category. Attributes like address, phone, etc. are considered secondary, so they are given $\frac{1}{3}$ of the weight in the similarity score function, while name, category and geographic proximity carry $\frac{2}{3}$ of the weight. The authors show top k matches for each entity to the user to decide.

3. Karam et al. [11] starts with filtering spatial entities that are 5 m apart. Then, the similarity of the name is measured with Levenshtein distance, the geographic similarity with Euclidean distance, and the keywords are compared semantically. In order to decide which pairs to match and which not, the similarities are fused using belief theory [25].

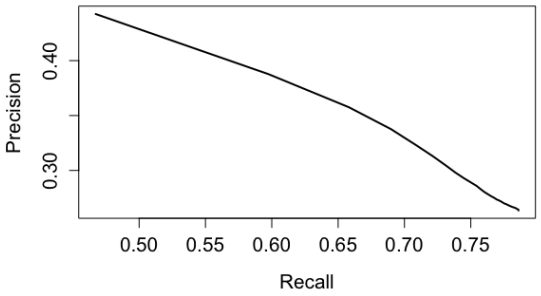


Fig. E.10: Performance of Morana et al. [13] on $\boldsymbol{D_{full}}$

The results using D_{full} and D_{sample} are presented in Table E.3. In general, all the methods performed better in the manually labeled dataset D_{sample} due to

the better quality of the labels. Berjawi et al.(V1) [12] has the highest *precision* of above 0.93, but a poor *recall* of at most 0.27 for both datasets and thus, a low *F-measure* of at most 0.43. It is actually expected, as Levenshtein similarity does not perform well with fields like address or phone, where a change in the digits makes a huge difference in the similarity of attributes. Berjawi et al.(V2) [12] yields reasonable results, the second best after *QuadSky*, with a *precision* of 0.73 in D_{full} and 0.97 in D_{sample}, and a *recall* of 0.56 in D_{full} and 0.60 in D_{sample}. The *F-measure* is 0.63 in D_{full} and 0.74 in D_{sample}.
To compare with Morana et al. [13], we tried all values k from 1 to the maximal matches for a single point. We plotted all combinations of *precision* and *recall* for different values of k in Fig. E.10 for D_{full}. The highest value of *F-measure* corresponded to a *precision* of 0.39 and a *recall* of 0.60. The behavior of Morana et al. [13] in D_{sample} is similar; the best value of *F-measure* was achieved for $k = 3$ and results are similar to those in D_{full}. The work of Karam et al. [11] achieves the highest *recall* of 0.73 but a very low value of *precision* of 0.23 for D_{full}. As a result, the *F-measure* is only 0.47. However, in D_{sample}, the method performs better overall (*F-measure* =0.6).
In comparison to all the baselines, QuadSky provides the best trade-off between precision and recall, and thus, the highest F-measure in both datasets. In the case of D_{sample}, *QuadSky* achieves the best *recall* compared to all baselines. The highest *precision* values for both datasets is achieved by Berjawi et al.(V1) [12] but a very low recall and poor model performance overall. In fact, models that achieve extreme values (high precision-low recall or low precision-high recall) are not a viable solution because they are either too restrictive or too flexible, and their predictability is poor. Most of the current work base their scoring function on the assumption that the geographical proximity is essential, so besides spatial blocks, they include Euclidean distance in their similarity functions. *QuadSky uses the spatial proximity for identifying candidates but not for making a decision*. Berjawi et al. [12] (V2) assumes the same weights for all similarities, and the reported values of *precision* and *recall* are reasonable. However, *the behaviors of the pairs can be of all types. QuadSky can capture these different behaviors better than a simple sum would*.

	D_{full}			D_{sample}		
Approach	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Berjawi et al.(V1) [12]	**0.93**	0.26	0.41	**1.00**	0.27	0.43
Berjawi et al.(V2) [12]	0.73	0.56	0.63	0.97	0.60	0.74
Morana et al. [13]	0.39	0.60	0.47	0.33	0.60	0.43
Karam et al. [11]	0.23	**0.73**	0.35	0.54	0.68	0.60
QuadSky	0.87	0.60	**0.72**	0.87	**0.82**	**0.85**

Table E.3: Comparison with the baselines

5 Conclusions and Future Work

Location-based sources provide rich information about spatial entities in terms of details and semantics. However, identifying which pairs of spatial entities refer to the same physical entity across different sources is a challenging problem due to the lack of labeled data, data quality problems in the sources and the difficulty of coming up with a data independent scoring function. In this paper, we addressed the problem of spatial entity linkage across multiple location-based sources. We proposed *QuadSky*, an approach that consists of two novel algorithms *QuadFlex* and *SkyEx*. *QuadFlex* arranges the spatial entities into spatial blocks with a low execution time (4-8 times less than Fixed Radius Nearest Neighbors algorithm [35] (FNN)) and a high percentage of comparisons (99.99% of FNN comparisons). *SkyEx* solves the data labeling problem using Pareto optimality and yields good results in terms of *precision* and *recall*. Moreover, due to its flexibility, *SkyEx* better captures the behavior of pairs and outperforms the existing baselines in terms of *F-measure*. More specifically, *SkyEx* achieves 0.84 *precision* and 0.84 *recall* on a manually labeled dataset and 0.87 *precision* and 0.6 *recall* on an automatically labeled dataset. In future work, we aim to study different blocking techniques that combine several attributes. Moreover, we plan to improve our proposed *SkyEx* in order to automatically select the k number of skylines based on the dataset characteristics.

References

[1] D. G. Brizan and A. U. Tansel, "A. survey of entity resolution and record linkage methodologies," *Communications of the IIMA*, vol. 6, no. 3, p. 5, 2006.

[2] K. Shu, S. Wang, J. Tang, R. Zafarani, and H. Liu, "User identity linkage across online social networks: A review," *Acm Sigkdd Explorations Newsletter*, vol. 18, no. 2, pp. 5–17, 2017.

[3] C. T. Yui, L. J. Liang, W. J. Soon, and W. Husain, "A survey on data integration in bioinformatics," in *International Conference on Informatics Engineering and Information Science*. Springer, 2011, pp. 16–28.

[4] D. Firmani, B. Saha, and D. Srivastava, "Online entity resolution using an oracle," *Proceedings of the VLDB Endowment*, vol. 9, no. 5, pp. 384–395, 2016.

[5] R. Maskat, N. W. Paton, and S. M. Embury, "Pay-as-you-go configuration of entity resolution," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXIX*. Springer, 2016, pp. 40–65.

[6] J. Efremova, B. Ranjbar-Sahraei, H. Rahmani, F. A. Oliehoek, T. Calders, K. Tuyls, and G. Weiss, "Multi-source entity resolution for genealogical data," in *Population reconstruction*. Springer, 2015, pp. 129–154.

[7] M. Edwards, S. Wattam, P. Rayson, and A. Rashid, "Sampling labelled profile data for identity resolution," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 540–547.

[8] O. Goga, H. Lei, S. H. K. Parthasarathi, G. Friedland, R. Sommer, and R. Teixeira, "Exploiting innocuous activity for correlating users across sites," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 447–458.

[9] A. Panchenko, D. Babaev, and S. Obiedkov, "Large-scale parallel matching of social network profiles," in *International Conference on Analysis of Images, Social Networks and Texts*. Springer, 2015, pp. 275–285.

[10] S. Isaj, N. B. Seghouani, and G. Quercini, "Profile reconciliation through dynamic activities across social networks," in *International Conference on Advanced Information Systems Engineering*. Springer, 2019, pp. 126–141.

[11] R. Karam, F. Favetta, R. Kilany, and R. Laurini, "Integration of similar location based services proposed by several providers," in *International Conference on Networked Digital Technologies*. Springer, 2010, pp. 136–144.

[12] B. Berjawi, E. Chesneau, F. Duchateau, F. Favetta, C. Cunty, M. Miquel, and R. Laurini, "Representing uncertainty in visual integration." in *DMS*, 2014, pp. 365–371.

[13] A. Morana, T. Morel, B. Berjawi, and F. Duchateau, "Geobench: a geospatial integration tool for building a spatial entity matching benchmark," in *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2014, pp. 533–536.

[14] X. L. Dong and D. Srivastava, "Big data integration," in *2013 IEEE 29th international conference on data engineering (ICDE)*. IEEE, 2013, pp. 1245–1248.

[15] P. Christen, T. Churches, and M. Hegland, "Febrl–a parallel open source data linkage system," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2004, pp. 638–647.

[16] O. Peled, M. Fire, L. Rokach, and Y. Elovici, "Entity matching in online social networks," in *2013 International Conference on Social Computing*. IEEE, 2013, pp. 339–344.

[17] G. Quercini, N. Bennacer, M. Ghufran, and C. N. Jipmo, "Liaison: reconciliation of individuals profiles across social networks," in *Advances in Knowledge Discovery and Management*. Springer, 2017, pp. 229–253.

[18] M. Magnani and D. Montesi, "A survey on uncertainty management in data integration," *Journal of Data and Information Quality (JDIQ)*, vol. 2, no. 1, pp. 1–33, 2010.

[19] R. Abdalla, "Geospatial data integration," in *Introduction to Geospatial Information and Communication Technology (GeoICT)*. Springer, 2016, pp. 105–124.

[20] P. Tabarro, J. Pouliot, R. Fortier, and L.-M. Losier, "A webgis to support gpr 3d data acquisition: A first step for the integration of underground utility networks in 3d city models," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, p. 43, 2017.

[21] S. Balley, C. Parent, and S. Spaccapietra, "Modelling geographic data with multiple representations," *International Journal of Geographical Information Science*, vol. 18, no. 4, pp. 327–352, 2004.

[22] V. Walter and D. Fritsch, "Matching spatial data sets: a statistical approach," *International Journal of geographical information science*, vol. 13, no. 5, pp. 445–473, 1999.

[23] M. Schäfers and U. W. Lipeck, "Simmatching: adaptable road network matching for efficient and scalable spatial data integration," in *Proceedings of the 1st ACM SIGSPATIAL PhD Workshop*, 2014, pp. 1–5.

[24] V. Sehgal, L. Getoor, and P. D. Viechnicki, "Entity resolution in geospatial data integration," in *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, 2006, pp. 83–90.

[25] A.-M. O. Raimond and S. Mustière, "Data matching–a matter of belief," in *Headway in spatial data handling*. Springer, 2008, pp. 501–519.

[26] G. Papadakis, E. Ioannou, C. Niederée, T. Palpanas, and W. Nejdl, "Beyond 100 million entities: large-scale blocking-based resolution for heterogeneous data," in *Proceedings of the fifth ACM international conference on Web search and data mining*, 2012, pp. 53–62.

[27] G. Papadakis, J. Svirsky, A. Gal, and T. Palpanas, "Comparative analysis of approximate blocking techniques for entity resolution," *Proceedings of the VLDB Endowment*, vol. 9, no. 9, pp. 684–695, 2016.

[28] H. Samet, "The quadtree and related hierarchical data structures," *ACM Computing Surveys (CSUR)*, vol. 16, no. 2, pp. 187–260, 1984.

[29] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8. Soviet Union, 1966, pp. 707–710.

[30] C. Fellbaum, "Wordnet," in *Theory and applications of ontology: computer applications*. Springer, 2010, pp. 231–243.

[31] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *ACL*, 1994.

[32] H. Köpcke, A. Thor, and E. Rahm, "Evaluation of entity resolution approaches on real-world match problems," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 484–493, 2010.

[33] Y. Censor, "Pareto optimality in multiobjective problems," *Applied Mathematics and Optimization*, vol. 4, no. 1, pp. 41–59, 1977.

[34] S. Isaj and T. B. Pedersen, "Seed-driven geo-social data extraction," in *Proceedings of the 16th International Symposium on Spatial and Temporal Databases*, 2019, pp. 11–20.

[35] J. L. Bentley, D. F. Stanat, and E. H. Williams Jr, "The complexity of finding fixed-radius near neighbors," *Information processing letters*, vol. 6, no. 6, pp. 209–212, 1977.

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-911-4

AALBORG UNIVERSITY PRESS