SYSTEMATIC APPROACHES TO IMPROVING GEOCODING AND REVERSE GEOCODING SYSTEMS

A Dissertation

by

ZHENGCONG YIN

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Daniel W. Goldberg
Committee Members,	Matthias Katzfuss
	Ruihong Huang
	Stacey Lyle
Head of Department,	David Cairns

December 2021

Major Subject: Geography

ABSTRACT

Geocoding, and reverse geocoding is a process that enables transitions between human-readable location information and machine-readable coordinates. The former converts human-readable information into machine-readable coordinates, and the latter does the opposite. These two processes perform an essential data-processing function that enables further spatial analysis to be conducted in a variety of fields such as public health. As a result, any subsequent studies and analysis that employ geocoded data are directly impacted by the quality of the output from geocoding and reverse geocoding systems. To be specific, there are three indicators of output quality: (1) match rate, which indicates the rate of successfully geocoded data and determines the usability of geocoded data for further analysis; (2) spatial accuracy, which enhances the validity of studies employing the geocoded data as input; (3) clear and concise metadata descriptions, which provides confidence for selecting the most relevant geocoded and reverse geocoded output.

In this work, the process of geocoding and reverse geocoding has been distilled to reveal the limitations of the existing solutions. The process of geocoding is divided into two sub-processes: (1) text retrieval, which aims to match an input description with a candidate of the highest textual similarity, and (2) geocoding interpolation, which corresponds to deriving the final output coordinates based on the geometrical and spatial attributes of the retrieved reference candidates. In examining these sub-processes, the limitations on the existing geocoding systems are identified as the incapacity for handling (erroneous) geocoding inputs, and the drawbacks of typical geocoding interpolation methods. As it relates to reverse geocoding, the sub-processes are: (1) match the most similar candidates to the respective human input and (2) re-rank the candidates according to specific criterion. The limitations of the existing reverse geocoding systems are the exclusion of topographical relationships amongst reference data, the ignorance of input uncertainty, and unclear metadata descriptions.

To overcome these limitations, three branches of research are conducted as follows. (1) To improve the robustness of geocoding systems for low-quality input, a set of parsing, matching,

and ranking methods are selected. To be specific, a unified evaluation protocol that is specific to geocoding text retrieval tasks (i.e., parsing, matching, and ranking) is defined. Next, a geocoding input dataset, which contains different degrees of errors and variants, is synthesized by mining human input patterns from existing geocoding transactions. From there, the input dataset is used to benchmark a set of geocoding parsing, matching, and ranking methods that are built upon Natural Language Processing (NLP) and Information Retrieval (IR) methods. (2) A novel geocoding interpolation approach, which incorporates Computer Vision (CV) technique, is developed to overcome the parcel homogeneity assumption made by the linear interpolation method; the parcel centroid assumption made by the polygon interpolation method, and the limited coverage of reference data used by the point interpolation method. (3) A new reverse geocoding ranking approach is introduced, which includes ranking output candidates by geometrical and topological attributes that are provided by the retrieved reference data, propagating input uncertainty to output, and fully quantifying each candidate based on relevance.

The work with these three branches aims to improve the match rate, spatial accuracy, and metadata descriptions of geocoding and reverse geocoding systems when facing low-quality input. Together, these improvements could lead to better geocoding and reverse geocoding systems through benefits gained in various spatial analyses and applications that use these systems as part of their data processing pipelines.

DEDICATION

To my family and all my friends. In memory of my grandfather and grandmother. This manuscript is written during the COVID-19 pandemic crisis. To everyone who fights to defeat it.

ACKNOWLEDGMENTS

There are many people I want to say thank you while I am writing this acknowledgment.

First and foremost, I would like to thank my advisor, Dr. Daniel W. Goldberg, who over the past many years has spent a significant amount of time and effort chiseling me into who I am today as a researcher. Without his constant patience, support, and advice, this dissertation would not have been possible. I still remember the moment when I stepped into his lab and became one of his students back in 2014. I thank him for introducing the geocoding world to me and allowing me to step away a little from the lab to explore the industrial world during the final stage of my Ph.D. studies. His enthusiasm for academia, passion for transforming research into tools and software everyone can use, and sense of humor have shaped me professionally and personally. Thanks, Dan.

I have been very fortunate to have Drs. Matthias Katzfuss, Ruihong Huang, and Stacey Lyle ion my committee. I received tremendous support and useful suggestions from them. Engaging in discussion with them or taking their courses gave me new insights for my research. I also want to thank Dr. Tracy Hammond, whose dedication to research and detailed review of my paper impressed me greatly.

I am very grateful to the great people I have bonded with while studying at Texas A&M University or in my life. I thank, Payton Baldridge, Aaron Harmon, and Edgar Hernandez for your help, when I was very new to the lab. I further extend my gratitude to my classmates and lab-mates: Andong Ma, Da Huo, Jinwoo Park, and Xiao Li; I hope you all can achieve your career goals. Furthermore, there are friends I made outside my academic department to whom I feel grateful. Yi Cui, Di Xiao, Jiayi Huang, and Shan Jin, I miss our after-work dinners and talk about various research/non-research topics together. Tiben Che, Shi Chang, Ye Wang, and Yicong Cai, thanks for making my life at College Station, Texas wonderful. Last, I would like to take this moment to thank Shengnan Liu, Weilei Qiu,and Min Yangthank Shengnan Liu, Wenlei Qiu, and Min Yang, who have given me lots of encouragement since we have known each other.

I have had the great fortune to do summer internships at Esri during my Ph.D. studies. My

thanks go to Jeff Rogers, Brad Niemand, and Jon Hancock, who give me the opportunity to engage with industry-level geocoding works for the very first time. Furthermore, I would like to thank Sathya Prasad, Peng Gao, and Ringu Abraham for providing me with opportunities to taste various GIS projects, and thanks, Jay Chen and Ethan Zhou, it's a great pleasure to work closely with you guys. During my time at Esri, I also met lots of great friends: Chong Zhang, who greatly impacted my Ph.D. studies and career, and Liwei Gao, who always encouraged me to do more coding exercises. To Tina, Dwayne, Vera, Zelin, and Jimmy, cheers to our friendships, hoping y'all keep moving to bigger and better things.

Last but never least, I am very grateful to my parents, my aunt, and my uncle for their unconditional love and support. They have always been there for me, from teaching me to wash my hands to complete this manuscript. I would never have gotten to where I am today without them. Additionally, I would like to thank Ming Yuan for the support and encouragement. I believe we will eventually see the light at the end of the tunnel.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Daniel W. Goldberg of the Department of Geography and the Department of Computer Science and Engineering, Professor Matthias Katzfuß of the Department of Statistics, Professor Ruihong Huang of the Department of Computer Science and Engineering, and Professor Stacey Lyle of the Department of Geography and the Department of Civil Engineering.

All work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported in part by funding provided by the North American Association of Central Cancer Registries, the US National Institutes of Health, and the US National Cancer Institute.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Research Motivation

Geocoding encodes human-readable location descriptions (e.g., postal addresses) to machine-readable geographic coordinates, and reverse geocoding converts geographic coordinates into location descriptions. These transitions between human-readable location information and machine-readable coordinates could help us discover insights behind spatial patterns. Recent decades have seen a proliferation of applications that employ geocoding and reverse geocoding techniques. These two services can be accessed anywhere, from academic spatial analysis to personal daily usage, from typing in search bars to voice-based question answering systems [1, 2, 3, 4]. As more human activities occur at a location, geocoding and reverse geocoding are starting to be used to support mining information that link with a particular site, known as location intelligence. These two services are becoming an essential step before conducting further spatial analysis in a variety of fields. Using epidemic investigation as an example, the first geocoding action occurred in 19th century as John Snow, who was one of the founders of modern epidemiology, plotted locations of cholera death cases on a map to identify the contaminated water pump [5]. Because of the development of geocoding and reverse geocoding services, we can do more in disease surveillance than before. By geocoding patients' residential addresses, we can monitor the disease outbreak situation, and we can track how diseases are transmitted [6]. By geocoding health provider location, we can conduct a spatial accessibility estimation for communities [7]. By reverse geocoding GPS coordinates, we can re-build people's trajectories for their activities and analyze their exposure risks to a disease [8]. During the recent COVID-19 pandemic, both geocoding and reverse geocoding have been actively used to monitor and track the outbreak situation [9, 10].

There are three requirements for geocoded data and reverse geocoded data to be useful in spatial analysis. The first requirement is spatial accuracy, which is considered the most important metric in output data quality [11, 12]. For geocoding data, spatial accuracy is the distance between

the geocoding output coordinates and the ground-truth coordinates. Errors in geocoded data are found to be propagated through to subsequent studies, thereby affecting the validity and accuracy of further data creation and the research conclusions of these studies [13]. For example, in air pollution exposure studies, two common geocoding error distances- 100 and 250 meters- could lead to biased exposure classifications [14], and spatial errors produced by street geocoding has been shown to result in consistent overestimation for the number of exposed individuals [15]. In a cancer risk study, a spatial error for residential locations lower than 50 meters is required to conduct further analysis [16]. For reverse geocoding data, spatial accuracy determines whether output location descriptions correspond to the input correctly or not. For example, a false positive location description returned by a reverse geocoding system may be located at the opposite side of the street, creating problems with passenger pickups as a driver may need to make a U-turn to arrive at the designated location.

The second requirement is the match rate. In both geocoding and reverse geocoding, match rate represents the fraction of successfully processed data. A recent study shows that 85 percent is considered the minimal acceptable match rate to conduct further analysis [17]. Therefore, match rate could determine the usability of the whole processed dataset (e.g., patients' residential addresses for a certain disease at a certain time period) for further analysis. Although match rates can be improved by reverting to a lower spatial accuracy area (i.e., from a street-level to a city-level) [18], it would largely sacrifice the spatial accuracy of geocoded data, which is unacceptable in practice. The third requirement is a descriptive metadata description for output geocoded and reverse geocoded data. For geocoded data, a high-quality metadata description should report its spatial accuracy, and an area that true geocodes are located at [19]. For reverse geocoding, output should be described by spatial topological relationships and how likely an output address description corresponds with input coordinates, with consideration of GPS errors. The quality of metadata descriptions becomes more important as it can facilitate confidence for candidate selection and candidate suggestion is demanded in more location-based service scenarios.

Because geocoding and reverse geocoding follow the "garbage in, garbage out" principle [20],

error-prone human input and inevitable GPS errors pose significant challenges for geocoding and reverse geocoding to meet these three requirements, once the corresponding process is not able to handle low-quality input properly. Given the impact of geocoded and reverse geocoded data quality on research investigations that employ them, improving the quality of output from these two systems when facing low-quality input becomes a necessity.

1.2 Problem Statement

Given the aforementioned wide usage of geocoding and reverse geocoding processes in different scenarios, substantial efforts have spent on development and evaluations of geocoding and reverse geocoding. Specific to geocoding, the scope of previous studies ranges from address parsing techniques [21, 22, 23]; feature matching techniques [24, 25, 26, 27]; interpolation methods [28, 29, 30]; output candidate selection and ranking [18, 31]; evaluation framework designs [19, 32] and impacts of output data quality on subsequent studies [33, 19]. For reverse geocoding, previous studies focused on reducing query latency [34, 35]; integrating more reference data source [36]; improving output candidate ranking quality [37, 38, 39, 40].

Despite this rich body of prior work, there are still gaps that need to filled to enhance geocoding and reverse geocoding systems. Since errors can be introduced at every step of the a geocoding process [20], instead of treating it as a black-box or end-to-end process, a geocoding workflow can be divided into two parts: (1) text retrieval, namely finding a candidate that has the highest textual similarity with respect to an input description, and (2) geocoding interpolation, which corresponds to deriving the final output coordinates based on the geometrical and spatial attributes of the retrieved reference candidates. For the text retrieval process, the inevitable challenge is erroneous human input. When geocoding systems can not handle errors and variants appropriately, the match rate and spatial accuracy of output data decrease, as geocoding interpolation could perform on the incorrect reference candidates. However, existing work related to the development or evaluation of geocoding systems seldom uses geocoding input that truly reflects the spectrum of human input errors, leaving the performance of systems uncertain when receiving user input in real scenarios. Moreover, due to the lack of a standard testing dataset and evaluation metrics, consistent

and reproducible geocoding evaluation results are difficult to obtain. For these reasons, it is hard to draw concrete conclusions about whether or not a geocoding or reverse geocoding system has been improved by directly comparing new experimental results to previous work.

For the geocoding interpolation process, either linear or polygon interpolation methods have to make assumptions to derive the final output, introducing significant spatial errors [28, 29]. Point interpolation, which generates highly accurate results [41, 42], is limited due to the unavailability of the rooftop reference dataset.

For reverse geocoding, when finding nearby candidates with input coordinates, most existing solutions use distance as the single criterion, failing to consider the topological relationships among reference data. This drawback can easily result in a situation in which while the returned candidate is the closest to the input points, it may be located on the opposite side of the street, making this candidate not ideal. Meanwhile, the current output metadata only includes the distance to the input and the spatial accuracy level of this candidate. There is no metric to describe how likely a candidate is to be the best, which limits confidence in selecting the best candidates. Input uncertainty (e.g., GPS accuracy) is also unavoidable, but ignored by existing reverse geocoding systems. As more mobile phones access reverse geocoding by using GPS input, taking input uncertainty into account becomes of importance.

1.3 Contributions of the Work

This research systematically studies the workflow of geocoding and reverse geocoding processes and attempts to improve the output quality of these two processes in a divide and conquer manner. Specifically, this work splits a geocoding process into two parts: text retrieval and geocoding interpolation. The first experiment targets geocoding text retrieval process, in which we detail a way to analyze existing geocoding user input and synthesize input queries that match the analyzed human input patterns (i.e., errors and variants that occur in different address components). Then, this experiment defines a unified evaluation protocol including testing dataset, evaluated methods, and metrics for evaluating three sub-tasks: parsing, matching, and ranking. Finally, a set of geocoding techniques that are robust to low-quality input are selected based on the benchmark

results. Experimental results produced by these benchmarks can serve as solid baselines for future geocoding system development. The second experiment aims to solve drawbacks of existing geocoding interpolation techniques and limitations of reference datasets. The proposed method also can be used to update address reference datasets. For reverse geocoding, the third experiment seeks to improve its output quality in three iterations. It first enhances the reverse geocoding workflow, leveraging distance and topological relationships among reference datasets to help candidate selection. We then quantify the distance and topological relationships for each candidate in a probabilistic manner. Finally, we propose a way to propagate input GPS uncertainty to output data, quantifying each candidate with its likelihood of being the best candidate.

In the end, geocoded data and reverse geocoded data produced in this work are expected to have better quality than existing approaches in terms of match rate, spatial accuracy, and metadata description facing low-quality user input. Point-by-point contributions of this work are summarized as follows.

- An automatic approach to detect and analyze errors and variants occurs in geocoding input from history data and synthesize low-quality geocoding input that mimics human input patterns.
- A unified geocoding text retrieval (i.e., parsing, matching, and ranking) task evaluation protocol including benchmark datasets, evaluation procedures and metrics.
- A set of geocoding text retrieval (i.e., parsing, matching, and ranking) techniques that are robust to low-quality input are systematically evaluated.
- An object detection based geocoding interpolation methods, which overcomes the typical drawbacks of existing interpolation techniques and the limitation of reference datasets used by geocoding systems.
- An automatic method to generate and update address or Point-of-Interest (POI) reference datasets using highly available remote sensing data.

- A reverse geocoding workflow that leverages spatial topological relationships between existing address models (i.e., address point, address parcel, street segment) for candidate selection, moving beyond simple distance-only measures to differentiate candidates.

- A reverse geocoding ranking approach to quantify topological relationships and distances of address candidates to input coordinates with a uniform quantification.

- A reverse geocoding algorithm for propagating input GPS uncertainties into output data.

1.4 Outline of the Dissertation

The remainder of this dissertation is organized as follows. **Chapter 2** details how to synthesize low-quality geocoding input data that mimic human input patterns and defines three benchmarks for evaluating three sub-task of geocoding text retrieval progress: Parsing, Matching, and Ranking, respectively. We assess the performance of a set of techniques for each sub-task when facing the synthesized low-quality geocoding input. **Chapter 3** aims to resolve the drawbacks and limitations of geocoding interpolation process. We describe how to integrate object-detection techniques from the Computer Vision domain into the typical geocoding workflow to overcome assumptions made by existing interpolation methods and to tackle the limitation of geocoding reference datasets. **Chapter 4** focuses on the entire reverse geocoding workflow. We elaborate on the proposed method to leverage and quantify distance and topological relationships among reference data so that output reverse geocoded candidates can be ranked in a quantitative manner. Meanwhile, we describe how to have reverse geocoding output candidate ranks taking input GPS errors into account, propagating such errors through the entire reverse geocoding workflow. **Chapter 5** concludes the dissertation by highlighting key findings and describing the potential for future work.

2. BENCHMARKS FOR GEOCODING PARSING, MATCHING, AND RANKING APPROACHES

2.1 Introduction and Related Work

Geocoding is the process of converting address descriptions into geographic coordinates [43]. Such transition has been widely used as a standard data processing step in various domains before conducting spatial analysis, and the quality of geocoding output is shown to be critical to subsequent studies which used geocoded data as input, as errors in geocoded data could easily propagate through the spatial analysis workflow [13].

Given the importance of geocoded data, substantial effort has been devoted to evaluating the performance of geocoding systems under different techniques and scenarios. These works include evaluations of output quality under different interpolation approaches (e.g., street linear interpolation, address polygon interpolation, address point interpolation, or object detection based interpolation), candidate matching methods (i.e., deterministic or probabilistic feature matching algorithms), fuzzy matching techniques (i.e., string similarity-based or dense vector similarity-based methods), land-use types (i.e., urban or rural areas), and input address types (i.e., postal address, road intersection, or P.O. Box) [29, 44, 45, 46, 25, 47, 48, 24]. In terms of geocoded data quality assessment metrics, most of these works quantify the geocoding quality by match rate and spatial accuracy [44, 46, 29]. Some frameworks evaluated also include how geocoded data impact the further spatial analysis operation such as point-in-polygon analysis [33] and how geocoding systems impact on an organization from the aspect of operations [19].

Despite this rich body of prior work, there are still some gaps that can fill in to facilitate better development and evaluations of a geocoding system. First, while every component in a geocoding system could introduce and propagate errors into final output [20], most of the existing work evaluate geocoding systems at a relatively Marco level, treating the entire geocoding process as a black-box or only assessing geocoding interpolation methods [44, 29, 49]. In fact, a geocoding

interpolation process, which derives the output point based on the geometry of a reference record, occurs after retrieving such a reference record (i.e., a feature matching process). Failures in a feature retrieval process are more likely to lead to a spatial error of final output severer than errors resulting from interpolation methods because the final output is derived from an incorrect feature record. However, few works [24, 50] have examined the quality of the feature retrieval process inside a geocoding workflow, and their evaluation metrics did not completely reflect its functionality in geocoding systems. For example, one should evaluate the ratio of retrieved matching candidates related to the entire reference dataset, as it is a necessity to pass a smaller number of candidates into the following ranking step to reduce latency for an entire geocoding process. Therefore, every step in a geocoding workflow needs to be evaluated individually, considering its ripple effects on other geocoding steps.

Second, testing data used in current work did not fully reflect the quality of geocoding input in real scenarios, making the performance of a geocoding system remains uncertain when facing (erroneous) input in reality. Since geocoding systems are known for following the "garbage in, garbage out" principle, and geocoding input is known as error-prone, containing both syntactic and semantic errors [51], low-quality input may pose challenges for geocoding systems to produce high-quality output. However, compared to the magnitude of errors can exist in geocoding input, testing input data in existing evaluation works contains relatively simple misspellings [51, 32], lacks certain types of errors (i.e., semantic errors) [32], or errors are generated based on other address system rules rather than United States one [25]. To better evaluate the performance of a geocoding system, testing input data should emulate human typing errors and should be closed to data that a system faces in real scenarios.

Third, datasets and evaluation metrics in current evaluation frameworks are not standardized [13], making evaluation results not comparable across different studies. In contrast, such standardized testing and annotated training datasets can be easily found in the Natural Language Processing (NLP) domain such as Named Entity Recognition (NER) tasks [52, 53, 54]. This can facilitate a new study as the experimental results that were produced by the same standard datasets can be

directly used as baselines for performance assessments. Recently, a study in the domain of Geographic Information Retrieval (GIR) targeted to help geo-parser selections by building a platform, which is configured with sets of geo-pasers, testing datasets, and evaluation metrics (i.e., recall, precision, and F-1 score) [55]. Given the heterogeneity of geocoding system configurations (i.e., parsing, database indexing, and ranking), an open-access and standardized dataset could largely facilitate the future development and evaluation of geocoding systems.

Given these observations, we argue that (1) testing input data that could reflect input quality in real scenarios should be utilized to better evaluate the performance of a geocoding system, (2) when facing low-quality input, the geocoding process which retrieves reference data becomes critical to determine the quality of output geocodes and should be evaluated in a fine-grained manner, and (3) testing datasets, metrics, and results of evaluation works should be standardized and open-access to serve as solid baselines for future development. Therefore, in this work, we present a benchmark framework to evaluate the geocoding text retrieval process. The advantages of the proposed framework can be summarized as follows.

- An data processing pipeline is developed to analyze human input address descriptions and synthesize low-quality geocoding input based on patterns of error and variations occurring in different address components.

- A unified evaluation protocol including evaluation dataset, evaluation procedures and evaluation metrics are defined individually for assessing three sub-process inside the text retrieval portion of a geocoding workflow: Parsing, Matching, and Ranking.

- A set of geocoding text retrieval techniques ranging from classic fuzzy string matching methods to recent neural network-based address parsing and document relevance and dense vector combined hybrid ranking approaches are systematically evaluated by synthesized address description input with different degrees of errors. The evaluation results could serve as baselines for future development and experimental comparisons as benchmark testing datasets and evaluation metrics have been standardized.

The remainder of this chapter is organized as follows. In Section 2.2, we provide the overview of the proposed benchmark, including benchmark goal, benchmark architecture, and basic setup. Section 2.3 introduces the approach to synthesize the low-quality geocoding input based on human search patterns. Section 2.4, Section 2.5 and Section 2.6 details the benchmark for the Matching, Ranking, and Parsing process of a geocoding workflow, respectively. In each benchmark, we define the benchmark task, evaluation data and evaluation metrics. We then elaborate the implementation details of each evaluated approaches and present evaluation results. We summarize the findings in Section 2.7.

2.2 Benchmark Overview

2.2.1 Benchmark Goal

When a geocoding input is received, a geocoding system needs to find one or more candidates with the highest textual similarity or are entirely identical to input queries. Once candidates are retrieved, geocoding interpolation methods can be performed on coordinates of retrieved candidates to derive final coordinates as geocoding output. We term this process as *geocoding text retrieval process*. We argue that the performance of such a process is of more importance than the performance of its successor workflow - the geocoding interpolation process in determining the quality of output geocoded data under low-quality geocoding input scenarios because any mistakes inside the candidate retrieval portion would result in a situation that geocoding interpolation algorithms are performed on completely wrong address records. To this end, the goal of the proposed benchmark is to assess and distill a set of geocoding text retrieval techniques that can be robust to error-prone human geocoding input queries to improve the final output quality of geocoded data.

2.2.2 Benchmark Architecture

The proposed benchmark is composed of four modules - (1) Address Error Injector; (2) Parsing; (3) Matching; and (4) Ranking. The first module describes how to synthesize input queries that match human input patterns. The three remaining modules correspond to the *Parsing*, *Matching*, and *Ranking* processes in a geocoding workflow, respectively. Figure 2.1 abstractly describes

the relationship between each module in this benchmark. Address Error Injector is mainly re-

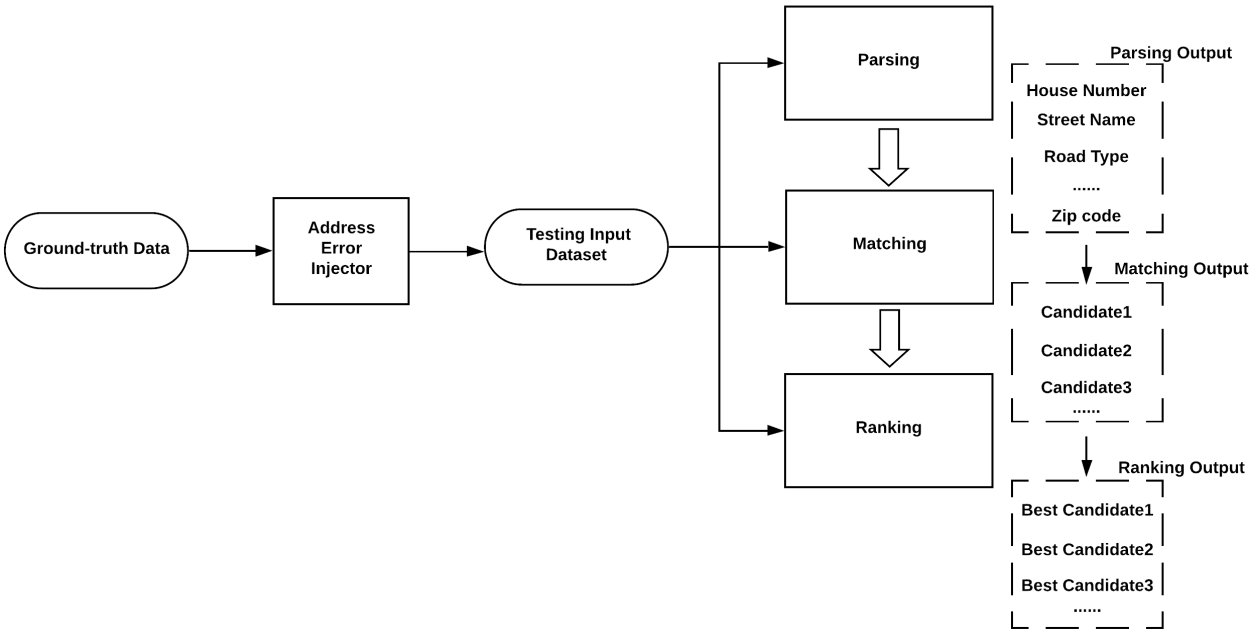


Figure 2.1: Benchmark Overview

sponsible for randomly injecting different degrees of errors and variants (using errors for short in this manuscript) to ground-truth geocoding inputs (e.g., address reference records) as needed so that the evaluated performance of a geocoding system could close to how a geocoding system performs when facing low-quality input in real scenarios. When feeding the synthesized low-quality input to a *Parsing* process, the entire input address description is split into several tokens based on the address component standards. Based on parsing results, a *Matching* process assembles query strings to retrieve a set of candidates, and finally, a *Ranking* process calculates a score for each candidate to reflect the textual similarity with respect to an input query. Given the strong connections among these three modules (from input to output), we design three different benchmarks to evaluate the performance of the *Parsing*, *Matching*, and *Ranking* processes individually for a geocoding system.

At a high level, each benchmark is designed to not only target the performance of a single

geocoding process but also considers how does this process impacts the remaining geocoding processes and final geocoding outputs. For example, the main goal of address parsing is to segment the entire input query into small chunks based on standard address elements such as house number, street name, and city name. Therefore, whether or not a parsing algorithm can split an input string into standard segmentation is the primary concern, given the scope of the parsing task only. However, with the consideration of the matching process, parsing results may not need to strictly stick to the standard address elements, as a relatively coarse-grained segmentation of input strings (e.g., house number and the entire street-level description) could have the same matching performance as the fine-grained segmentation. Thus, each benchmark attempts to answer the following questions: (1) which method could achieve the best performance given the task itself only, such as an address parsing algorithm that achieves the best F1 score, and (2) which method could lead to the best performance considering the impact of output from the current task on other tasks. In terms of evaluated approaches, we select a set of popular or state-of-the-art approaches and techniques from studies in related domains. For instance, the evaluated address parsing techniques include recent advanced neural network-based parsing methods from NER tasks [56], the ranking approach involves dense vector-based multi-staged ranking methods [57]. As for matching techniques, we evaluate a set of classical fuzzy string matching techniques that are widely used in various entity matching works [24, 48, 58]. To this end, the evaluation results could reflect how does a geocoding system that employs these advanced techniques performs facing low-quality input.

2.2.3 Benchmark Configuration

Data. We choose Navteq 2016 address point reference datasets as the geocoding reference dataset to retrieve candidates for both matching and ranking processes. The testing input dataset used for different evaluations is randomly selected from this reference dataset. To ensure the heterogeneity of address formatting, we randomly extract address reference data based on unique street base names from each unique pair of city and zip codes in every state in the United States, resulting in 60,483 unique address descriptions. These sampled data serve as ground-truth data for the corresponding task, and each address record will be paired with itself with injected random

errors and variants on any address components.

Software. Elasticsaerch, a NoSQL-based text search engine is selected as the data warehouse in this benchmark, as it provides flexible storage schema setup options [59], mature relevance ranking scoring function [60], and horizontal scaling capability [31].

Hardware. All benchmark evaluations are conducted on a workstation equipped with 16 cores 2.60-GHz Intel(R) Xeon(R) E5-2670 CPU, 64 GB DDR3 1600-MHz RAM, and SSD for storage.

2.3 Address Error Injector

Address Error Injector (AEI) is designed to randomly inject different errors into standard address records so that synthesized address descriptions, which mimic human input patterns, could be used as low-quality geocoding input for various evaluation tasks.

To capture human input patterns, we extract three-month geocoding transactions from Texas A&M geocoding platform[1] and only keep these inputs which cannot lead to full matching scores (i.e., the input and the reference data are not completely identical.) In total, we obtain roughly 30,000,000 input queries. Next, we iterate each input and its corresponding reference data to detect a set of possible input errors. Since address reference records are already segmented based on address components, errors can be easily detected by aligning input address descriptions with their corresponding description in address reference datasets. These detected errors are known as syntactic errors [32, 51] and can be generated by reversing the process of how these errors are detected. For example, by aligning address reference records and user input, we can detect an error type: Spanish prefixes omission and collect a set of commonly used Spanish prefixes in street names with the help of domain knowledge. When injecting the error of Spanish prefixes omission, we can utilize the collected Spanish prefixes dictionary to identify Spanish prefixes existing in street descriptions and delete such prefixes. As for typographic errors, we replace one character of a word with one of its nearby characters based on the keyboard position, the exact mechanism that is used by Freely Extensible Biomedical Record Linkage (FEBRL) [32].

The other category of errors that could happen in geocoding input queries is semantic errors

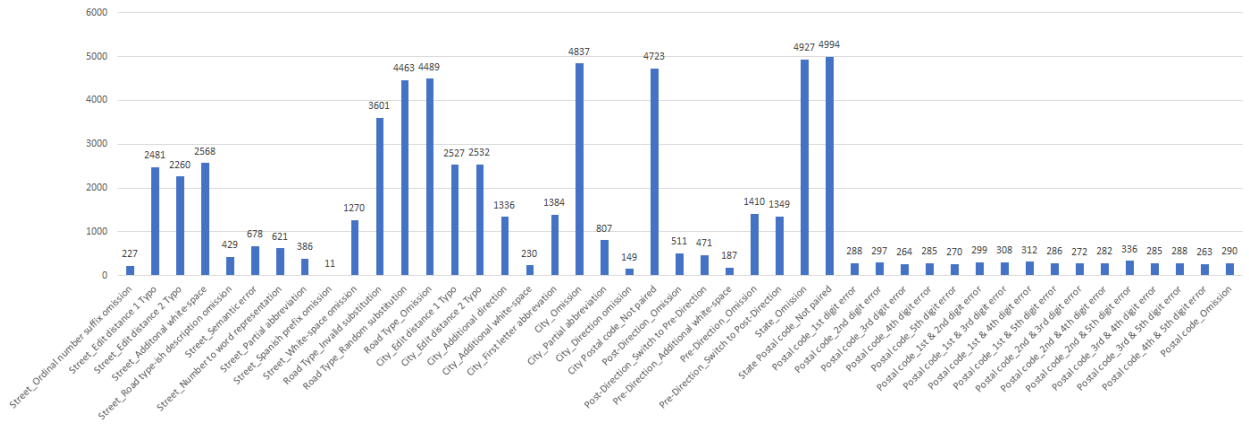
[1] `https://geoservices.tamu.edu/`

[51]. Compared to syntactic errors, semantic errors are relatively hard to detect from the log history because it is hard to use a threshold to determine if two terms have a similar meaning. To reproduce semantic errors, we utilize the WordNet corpus[2] from the NLTK library [3] to find words with similar meaning as substitution. Specifically, we first give each token a Part of Speech (POS) tag, and then we tend to find a synonym with the same POS tag as the current token. If more than one token has a valid synonym substitution, we prioritize an adjective token more than a noun token to complete the substitution process. If no suitable synonyms are found, we try to iterate a lemmatization of the current token and use it as a substitution. In total, we are able to inject 45 different errors occurring in all possible address components into an address description. We group these errors on the basis of address components and provide an example for each error as a reference in Table 2.1. To simulate geocoding input with various quality [50], AEI can synthesize address descriptions containing 1 or 2 degrees of input errors by randomly selecting one applicable of input errors into one of the address components. When 2 degrees of errors are desired, AEI injects two errors into two distinct fields. Figure 2.2 describes the distribution of each injected error after randomly injecting 1 or 2 errors into the sampled testing dataset described in Section 2.2.3. We denote these two synthesized datasets as 1-error testing input dataset and 2-error testing input dataset in this manuscript.

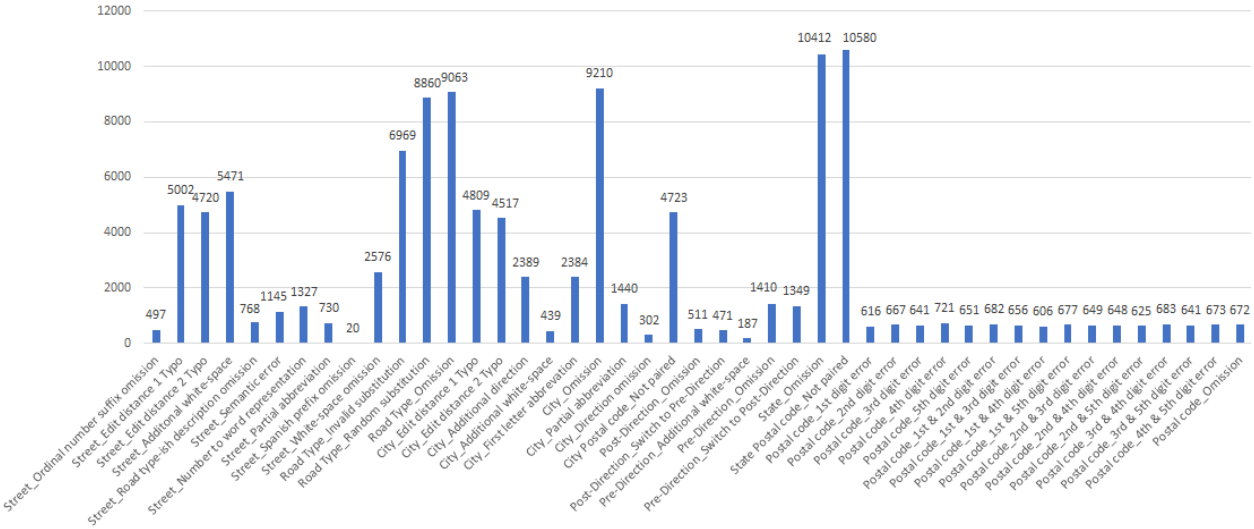
2.4 Benchmark for Geocoding Matching Process

In a geocoding workflow, a *Matching* process plays a pivotal role in determine output quality, as such a process assembles query strings based on different entity resolutions of parsing results and looks up candidates that will be fed into the final output ranking process, thereby profoundly deter-mines if correct candidates could be included in the output list of a geocoding text retrieval process. Additionally, a matching process primarily impacts the latency for a single geocoding process, as different matching strategies will incur various comparisons to be conducted to find likely matched records as output. Therefore, the overall goal of this benchmark is to evaluate a

[2] https://wordnet.princeton.edu/
[3] https://www.nltk.org/



(a) 1-error



(b) 2-error

Figure 2.2: Distributions of randomly injected errors for the 1-error and 2-error testing input datasets

set of techniques that allows a matching process to retrieve a reasonably small group of candidates that include the ground-truth candidate while keeping the whole candidate searching process to be finished in a relatively short amount of time.

2.4.1 Blocking Fields Selection

In the record linkage domain, blocking fields are typically derived from the attribute of reference records. As a results, "similar" records that share the same Block Key Value (BKV) can be grouped together [61]. To retrieve matching candidates, string comparisons must be made be-

Table 2.1: Input error and variations on each address component

Address Element	Error Type	Error Example
Pre-/Post-Direction	Typo	Norh West → **Norh** West
	Omission	**East** Main St → Main St
	Partial abbreviation	North West Main St → **N** West Main St
	Additional white-space	NorthWest Main St → **North West** Main St
	Pre/Post-Direction switch	E Main St NW → **NW** Main St **E**
Street Base Name	Edit distance 1 Typo	Main St → **Man** St
	Edit distance 2 Typo	Main St → **Mian** St
	Ordinal number street suffix omission	5th Ave → **5** Ave
	Spanish prefix omission	**La** Brea Ave → Brea Ave
	White-space omission	Memory Hill → **Memoryhill**
	Additional white-space	Reachcliff → **Reach Cliff**
	Partial abbreviation	Warm Mountain → Warm **Mtn**
	Road type-ish description omission	Camden Cove Pkwy → Camden Cove
	Number to word representation	5th Ave → **Fifth** Ave
	Semantic error	Small river St → Small **creek** St
Road Type	Omission	Main **St** → Main
	Road type random substitution	Main St → Main **Ave**
	Invalid road type substitution	Main St → Main **St St**
City Name	Edit distance 1 Typo	Austin → **Austiun**
	Edit distance 2 Typo	Luverne → **Luvre**
	Additional direction	Houston → **South** Houston
	Direction omission	**North** Little Rock → Little Rock
	Partial abbreviation	Saint Maries → **St** Maries
	First character abbreviation	Los Angeles → **LA**
	City Postal code not paired	College Station, 77845 → **Byran**,77845
	Additional white-space	Northgate → **North Gate**
	Omission	**College Station**, TX 77845 → TX 77845
State Name	Omission	Houston, **TX** 77001 → Houston, 77001
	State Postal code not paired	TX 77845 → **TN** 77845
Postal Code	Omission	Houston, TX **77001** → Houston, TX
	one of any digit mismatch	77845 → **77843**
	two of any digit mismatch	77845 → **76443**

tween the input and each record in the reference dataset. Thereby, blocking groups have smaller dimensions in relation to the dimensions of the entire reference dataset, can potentially increase the efficiency of the retrieval process [48]. An empirical study for several linkage record systems have shown that the selection of blocking fields is more important than the matching approaches in terms of determining the matching performance [62].

With regard to geocoding systems, each reference record is an address description and the blocking fields could be selected from the base of address components, as shown in Figure 2.3.

Such selection can impact a geocoder from two perspectives. First and foremost, it could determine

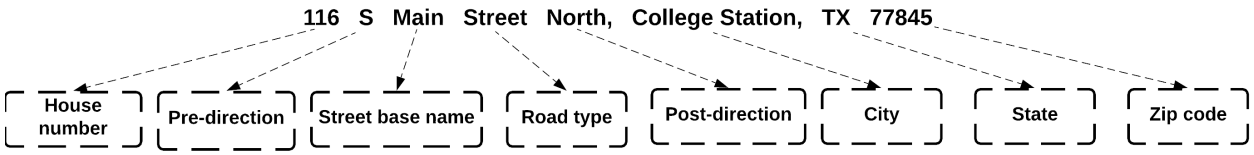


Figure 2.3: Address components of an address description

the search performance for an input query. Different blocking strategies (e.g., addresses grouped by city names vs. addresses grouped by state names) could result in various dimensions, directly influencing the latency of search queries. Secondly, the choice of blocking fields also creates some extent of relaxation space when searching a matched candidate. For example, for an input address with omitted street directions, querying against the direction field may lead to zero matches. On the contrary, querying without this field could lead to a match to proceed as long as at least one of the other reference features (e.g., street name and city name) have a match. Since a complete postal address has geographic partitions naturally, separated by states, cities, and zip codes. As a result, modern geocoding systems usually employ these administrative areas as BKVs to group addresses [63, 24, 2]. We follow the same notion to use these administrative areas directly as BKVs. In terms of street-level descriptions, we only use the street base name from Figure 2.3 as a BKV for two reasons: (1) to maintain the spatial accuracy of geocoded output, we only attempt to retrieve candidates that at least match to the same street block as long as a street name is presented in the input, and (2) given the fact that errors could occur in regard to street direction and road type, matching the street base name could tolerate errors that occur with these street-level address components.

It is worth noting that the prerequisite to using these aforementioned BKVs is that the whole address is split into small segments correctly by an address parser according to these BKVs. Compared to city and zip code, extracting a street base name from input queries is relatively difficult; one must correctly identify the boundary of street direction and road type. Meanwhile, the street

base name itself has many ambiguities, and it contains various numeric, person or geographic descriptions. In contrast, city and zip code in input queries are more likely to be identified with the help of gazetteers, lexicon characteristics (e.g., five digits), relative positions. To cover the possible coarse-grained parsing outcome, we defined two additional BKVs: (1) long street-level description, namely the description between the house number and city name fields, and (2) the short street-level description which is composed of street base name and road type only. These two BKVs can be used in the event that an address parser cannot correctly identify the boundary of certain street-level components (i.e., street directional, street base name, road type).

In summary, we use the following BKVs in the further matching process evaluations:

- Fine-grained BKVs: street base name; city; zip code;
- Coarse-grained BKVs:
  - Long street-level descriptions: street direction + street base name + road type;
  - Short street-level descriptions: street base name + road type.

2.4.2 Fuzzy Matching Approaches Evaluation

2.4.2.1 Fuzzy Matching Approaches

Only allowing strict matching strategies to query against these BKVs may lead to low match rates (i.e., no candidates can be returned) or lower spatial accuracy output (e.g., revert to larger administrative areas such as postal code or city level areas) in the event that the input contains errors. For this reason, a variety of approximate string matching approaches have been developed and applied to overcome the mismatch between input queries and reference features in data linkage or information retrieval systems [48]. We select a set of matching techniques from three major method categories (i.e., phonetic-, character-, and token-based methods) because of their performance on matching entity names in various studies [48, 64, 24, 65] as follows.

- Phonetic-based methods

Similar to all phonetic-based approaches, the selected **Soundex** and **Double-Metaphone** encode the string based on its pronunciation in English. Soundex has been widely implemented in geocoding systems; this approach, however, is only applicable to the English language and is sensitive to the position that a typo may occur [49]. As opposed to Soundex, Double-Metaphone encodes certain groups of characters in a string, and it returns two phonetic encodes that are composed solely of alphabetic letters [66].

- Character-based methods

  Edit distance calculates the number of steps in required edit operations (e.g., insert, delete, substitute, and swap) to convert one string into another. There are two ways to leverage the calculated edit distance to find matching candidates. One method is to convert the edit distance into a similarity score between [0, 1] using the following equation:

$$\text{Similarity} = 1 - \frac{Edit(s1, s2)}{\max(|s1|, |s2|)} \tag{2.1}$$

  where $s1$ and $s2$ are the two strings to be compared, and $Edit(s1, s2)$ calculates the edit distance between $s1$ and $s2$. $Max(|s1|, |s2|)$ obtains the shorter length of these two strings $s1$ and $s2$. We term this method as **Edit Similarity**. The other method entails using a threshold to include all candidates with an edit distance that is less than the aforementioned threshold. In this benchmark, we only evaluated the threshold of 1 or 2, denoted as **Edit Dist T1** and **Edit Dist T2**, as errors with an edit distance of 3 are relatively rare in practice.

  Jaro and **Jaro-Winkler** belong to the same family of approximate matching techniques, and were originally designed for matching people's names [65]. Jaro distance calculation combines q-gram and edit distance techniques by tracking counts of the same characters between two strings and the number of transpositions that exist in two compared strings. The Jaro-Winkler method is an extension of the Jaro distance calculation. It weighs more scores for matching candidates that share more prefixes with an input source string under the assumption that errors tend to occur less frequently at the beginning of a string.

***q*-gram** divides the whole string into multiple sub-strings with a length of q; this allows for the similarity to be measured in the notion of the number of common substrings. Typically, q of of 2 or 3—known as Bi-grams and Tri-grams—are the most common cases in string approximate matching studies. A common variation of q-gram is to add (q-1) special characters to the original string, which has been proven to be robust to typographical errors in a study [67]. The generated grams are used to select candidates in two ways: (1) select candidates that share at least n grams and (2) select candidates based on calculated similarity. For the first method, we initially process reference strings by utilizing q-gram methods with padding; this increases the chances that two strings will share the same grams. These reference strings are subsequently grouped by each substring obtained by q-gram methods in a look-up dictionary-as originally proposed by [68]-and track the number of q-gram substrings shared with the input string for every candidate. Those that have more than n shared grams are retrieved. We name each variation of this method in the format of Pad q-grams Tn. For example, Pad 2-grams T1 means the padded version of 2-grams with a threshold of 1. For the second method, we select the Dice coefficient to quantify the string similarity, as this quantification was reported in a study to have the best matching quality for matching street and town names in the Netherlands [24]. Given substrings generated by q-gram, Dice coefficient uses the number of common substrings to divide the average number of two compared substrings. We evaluate the Dice coefficient calculated by 2-grams and 3-grams of substrings, in reference to [24], and are denoted as **2-grams Dice Coeff** and **3-grams Dice Coeff**, respectively.

- Hybrid methods:

  Given the fact that string similarity calculation is computationally expensive when used to speed up the search process, [24, 68] attempted to filter out candidates that has less than certain number of padded n-grams first; subsequently, the similarity calculation was conducted. To reduce the number of combinations, we select the most efficient filter-based method variation first (i.e., the granularity of word grams and threshold of shared grams)

and subsequently combine it with similarity methods that have higher match rates than the selected filter method..

- Token-based methods:

  Term Frequency-Inverse Document Frequency (TF-IDF): Compared to character-based approaches, token-based approaches focus on using multiple sets of tokens extracted from the whole input string for similarity calculation. **TF-IDF** is the most widely used method in the information retrieval domain. This method considers a token as important if it appears frequently within a single document or appears infrequently within the whole document collection. In this benchmark, we use Elasticsearch to calculate the relevance score based on TF-IDF, as we already use Elasticsearch to index the reference dataset. Elasticsearch provides a default scoring function, BM25, which is an enhanced TF-IDF for matching long textual descriptions. Hence, we also employ such a variation to calculate the relevance score. [69], we also employ such a variation to calculate the relevance score. **TF-IDF | BM25** indicates a relevance score and is calculated by a TF-IDF or BM25 function, respectively. A token may also be processed by a q-gram method. We compare scores calculated by two different token resolution Bi-gram and Tri-gram tokens, denoted as **2-grams** and **3-grams** and compared to scores calculated by individual terms. These three tokens resolution can have two versions- padded version and non-padded version. For example, (Pad) 2-grams TF-IDF represents scores calculated by TD-IDF with a padded and non-padded version of the 2-grams token. Given the fact that Elasticsearch provides a fuzzy query option based on edit distance methods, we also include fuzzy queries to see if the matching quality can be further improved. It is worth noting that since Elasticsearch cannot handle the errors at the first position [59], we generate a padded version of tokens by adding a special character and only evaluate scores calculated by this padded version of tokens. In summary, the evaluated token-based method variations depend on (1) the scoring function, **TF-IDF | BM25**; (2) the token resolution, **2-grams** and **3-grams**; (3) tokens with and without padding, **(Pad)**. (4) usages of fuzzy queries, **Fuzzy**.

These matching methods can be grouped into different streams depending on how candidates are produced as well as the usage for matching BKVs, as shown in Table 2.2. The first category is filter-based approaches; these approaches select candidates once they meet certain criteria, such as the same Soundex-encoded string. Therefore, one can use these approaches as coarse-grained filters to reduce the computation expense for further selections. In contrast, the second category is similarity-based approaches; these approaches compare the input string with every record in a reference dataset, assigning each record a similarity score via a string similarity quantification and use it to match. These scores could potentially be re-used in the further ranking steps, such as producing weighted matching scores or classifying matching status [25]. A hybrid method leverages the advantages of filter-based methods and similarity-based methods. This category of methods attempts to use filter-based methods to filter out some unqualified candidates and calculate a similarity score to each candidate. In this way, candidates can be ranked in a quantified manner. We separate token-based methods with all other methods. Token-based methods, which are design to match documents with a long text length, are used for coarse-grained BKVs. In contrast, the remaining methods are used to match fine-grained BKVs, which has the relatively short text lengths.

2.4.2.2 Data and Evaluation Metrics

Data

Since the performance of matching techniques is evaluated on the basis of BKVs, datasets used in this benchmark are specific to these BKVs. Overall, the selection of data follows the principle that input data are extracted from an address reference dataset and should have wide coverage of various address description formats (e.g., numeric, fraction, and single-letter). The procedure to prepare data for each BKV is described as follows.

Zip code: We compile a list that contains 30,646 zip codes in the United States from Navteq2016 address point reference datasets[4] and randomly inject all possible errors occurring in the field of zip code summarized in Section 2.3. We exclude errors that result in more than three different digits because such errors indicate that the input zip code and the input state are not paired, which

[4] `https://www.here.com/navteq`

Table 2.2: Category of evaluated matching techniques

Category	Matching Techniques
Similarity-based	Jaro Jaro-Winkler Edit Similarity 2-grams Dice Coeff 3-grams Dice Coeff
Filter-based	Soundex Double-Metaphone Edit Dist T1 Edit Dist T2 Pad 2-grams T1 Pad 2-grams T2 Pad 3-grams T1 Pad 3-grams T2
Hybrid	Similarity based methods + Pad q-grams
Token-based	(Pad) TF-IDF ǀ BM25 (Pad) 2-grams TF-IDF ǀ BM25 (Pad) 3-grams TF-IDF ǀ BM25 Fuzzy Pad TF-IDF ǀ BM25 Fuzzy Pad 2-grams TF-IDF ǀ BM25 Fuzzy Pad 3-grams TF-IDF ǀ BM25

should be handled by a matching logic rather than handled by fuzzy string matching techniques.

Street base name and **City**: We first extract and obtain a unique list of street and city names in the United States from Navteq2016 address point reference datasets. Then we order the unique street names alphabetically and randomly sample one street name per 10 address records, resulting in a unique street name list that has 76,025 street names. Using a similar producer, we compile a list of 34,934 unique city names. Then we inject various applicable errors described in Section 2.3 to each street and city name in the compiled lists, meanwhile, each original address description is used as the ground-truth data. To this end, every ground-truth address description is automatically paired with synthesized erroneous formatting.

Street-level descriptions: We randomly select 58,357 address records from Navteq2016 datasets with different states and zip code pairs across the United States. Then we randomly inject various

errors to street-level address components (i.e., street direction, street base name, and road type), and finally, we assemble these address components to get two datasets based on the definition of short and long street-level descriptions, as described in Section 2.4.1.

Evaluation Metrics

When retrieving the matching candidate to an input, each fuzzy matching technique attempts to generate a set of candidates with respect to this input, so that the searching criteria could be constructed and executed from these candidates. Thus, an ideal matching technique should be able to (1) generate a relatively small number of candidates, because more candidates require more comparisons to be conducted later in the ranking stage; (2) give a high-ranking position to the correct reference candidate that matches the input, as a higher ranking position could help to cut off several top candidates to build up queries; and (3) complete the candidate-searching process in a relatively short time period. To this end, we evaluate the performance of each matching technique primarily through by two metrics: match rate and match efficiency. Match rate, which measures how likely a matching technique could find the ground-truth data, is more important for differentiating different matching techniques. This is because whether the candidate list includes the ground-truth data will directly determine whether the entire geocoding process could output the correct geocoded data. For matching efficiency, we quantify this metric in two ways depending on how matching candidates are selected.

$$\text{Match Efficiency} = \begin{cases} \text{Filter Ratio} = \frac{\text{\# of candidates}}{\text{\# of total reference dataset}} & \text{Filter-based methods} \\ \text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank}_i} & \text{Similarity-based methods} \end{cases} \tag{2.2}$$

When candidates can be ranked by similarity values, we quantify match efficiency by the Mean Reciprocal Rank (MRR), which indicates the average rank position for all ground-truth data. If candidates are selected by filter-based approaches, we quantify match efficiency by the ratio of the number of matching candidates related to the number of address records in the reference datasets, which reflects how many reference records are filtered out. We report runtime measurements for a matching process under the scenario that the search space comprises the entire reference dataset,

which indicates the cost-time for the worst searching case where no additional information, such as zip codes or state names can be leveraged to shrink searching space.

2.4.2.3 Evaluation Procedures and Results

In this section, we describe the evaluation procedures, present and discuss assessment results for each BKV listed in Section 2.4.1

Zip code: Unlike other BKVs, zip codes in the U.S. address system are completely composed of digits and digit positions indicate whether or not addresses belong to the same administrative area. For example, if the zip code of two areas shares the same first two digits, these two areas are located in the same state. Previous works have employed the relationship between zip codes and state or the first-half prefix to filter zip codes not belonging to the same region as input [70]. Arguably, such methods are not robust since errors occur at the first-half portion of a zip code. Thus, we consider that the best way to find matching zip code candidates is to generate a set of zip code candidates that share certain portions with the input one. To this end, we search zip code candidate using the q-gram based filter method described in Section 2.4.2.1. Depending on the method to obtain q-gram and the threshold of shared chunks to consider as a match, we denote these methods as *q-grams* T*n* and *Pad q-grams* T*n*, respectively. We also compare the performance of the match approaches that leverage the first two and three digits of zip codes to conduct a candidate look-up, denoted as First 2-digit and First 3-digit, respectively.

Table 2.3 summarizes the performance of each matching approach. As expected, both First 2-digit and First 3-digit have relatively low match rates in comparison to all other q-gram based approaches, as they assume that no errors occur inside the first n-digits, which is not practical in real scenarios. Padded q-grams methods have higher match rates than q-grams without padding, because padded zip codes could result in more chunks than non-padded ones, creating more chances to find the common chunks between input and reference zip codes, especially the correct digits are only located at the first and last positions. Within the same category of matching approaches, match rates decrease as the increasing threshold for the number of common chunks that are used for considering two zip codes is a match. This is expected because the increased threshold requires

the input and reference zip code to be more identical; therefore, most zip codes will be filtered out. Overall, Pad 2-grams T2 has the best performance compared with all other approaches, as such an approach could always retrieve a list of candidates that contains the correct zip code while keeping the candidate list small, regardless of errors in the input zip codes. By using this method we can conduct an exact matching strategy to find zip code candidates. Such a matching strategy is computationally efficient, since the reference datasets could be pre-processed by the q-gram based method during an indexing phase. Such that, we do not need to employ these fuzzy string similarity methods as these methods are time-consuming and do not need to fine-tune the similarity quantification for every digit as digits in the leading positions are of more importance than digits in the tailing positions in terms of determining the administrative areas.

Table 2.3: The performance of matching techniques for zip codes

Matching Approach	Match Rate (%)	Match Efficiency
Pad 2-grams T1	100	20.84
Pad 2-grams T2	100	3.20
Pad 2-grams T3	60.10	0.41
Pad 2-grams T4	33.48	0.05
Pad 3-grams T1	93.46	11.23
Pad 3-grams T2	66.79	1.25
Pad 3-grams T3	40.20	0.14
Pad 3-grams T4	6.585	0.01
2-grams T1	93.15	3.92
2-grams T2	53.31	0.32
2-grams T3	13.13	0.02
3-grams T1	33.14	0.32
3-grams T2	6.60	0.01
First 2-digit	40.18	1.05
First 3-digit	20.07	0.11

Street base name and **City**: The way to evaluate the best matching techniques for street base names and city names are similar. First, we apply the similarity- and filter-based methods listed in Table 2.2 to the defined testing dataset and evaluate their performance using defined metrics. Since

similarity based matching techniques in Table 2.2 will loop through the entire reference dataset to calculate a similarity value for each record, the match rate is always equal to one, making this metric meaningless when comparing to the filter-based methods. Meanwhile, a low similarity threshold leads to more candidates being returned by a matching process, potentially reducing the efficiency of the matching and ranking process. Therefore, we empirically select a similarity score of 0.5 as the threshold, namely, only candidates with similarity values larger or equal to 0.5 (denoted as simi >= 0.5) will be considered as a match. This threshold is determined based on distributions of similarity values for both street base names and city names. Second, we choose the most efficient filter-based and similarity-based methods that have higher match rates than the selected filter-based method to compose hybrid methods and conduct assessments. Since retrieved candidates are eventually utilized for building query conditions by the logic of *OR*, we evaluate the overall performance of each hybrid method variation with the retrieval of the at-most first 30, 50, 100, and all candidates to see whether it is possible to reduce the number of retrieved matching candidates.

Street base name: We first group match rates by error types to get a notion of which matching techniques should be used to handle each error type. Table 2.4 lists the match rate of each similarity-based method and filter-based method. As mentioned before, to speed up the search progress (i.e., reduce the number of comparison between input and reference data), we use a threshold of 0.5 to consider a similarity based method successfully identify the correct ground-truth data for an input. As shown in Table 2.4, among all listed error types, these matching approaches systematically have relatively low match rates on the Error V - "Number to word representation". Under this type of error, a majority of cardinal number-based street names are converted to their word representations. For example, 100th street is converted to one hundred street. As these matching approaches are designed to handle partial mismatching between two strings rather than dealing with two strings with large differences, low match rates under this type of errors are expected, suggesting that this type of error should be handled in a different way. For the remaining error categories, similarity-based approaches consistently have a higher match rate than filter-based

Table 2.4: Match rate of matching techniques under different street name error categories

Match method \ Error category	I	II	III	IV	V	VI	VII	VIII	IX
Similarity-based Approach (simi >= 0.5)									
Jaro	99.98	99.95	100	99.34	13.64	99.77	95.59	99.99	100
Jaro-Winkler	99.98	99.95	100	99.70	14.84	99.96	95.59	99.99	100
Edit Similarity	99.99	99.62	100	89.36	1.00	99.85	98.90	99.99	100
2-grams Dice Coeff	99.44	86.86	99.78	76.80	1.33	96.89	100	99.83	100
3-grams Dice Coeff	89.87	54.98	96.35	72.59	1.07	93.35	99.45	99.62	95.78
Filter-based Approach									
Soundex	66.41	43.19	93.22	47.07	0.42	65.73	14.36	96.08	0
Double-Metaphone	59.26	36.76	90.40	43.60	0.42	62.54	14.36	94.89	0
Edit Dist T1	99.99	0	100	17.74	0.23	0	0	99.98	0
Edit Dist T2	100	99.97	100	20.75	0.49	0.50	0	99.98	100
Pad 2-grams T1	97.31	96.98	99.28	99.81	46.79	99.85	100	99.52	100
Pad 2-grams T2	97.26	96.42	99.28	99.81	29.87	99.81	100	99.52	100
Pad 3-grams T1	97.31	96.83	99.28	99.81	14.55	99.85	100	99.53	100
Pad 3-grams T2	97.26	95.95	99.28	99.81	5.18	99.81	100	99.52	100

* Error category: (I) One edit distance typographical error; (II) Two edit distance typographical error; (III) Additional white-space; (IV) Semantic error; (V) Number to word representation; (VI) Partial abbreviation; (VII) Spanish prefix omission; (VIII) White-space omission. (IX) Ordinal number suffix omission.

methods, as every address records in the reference datasets receive a score, whereas, filter-based methods drop address records once certain criteria fail to be satisfied. For example, Edit Dist T1 only considers candidates that have an edit distance less than 2 to the input string, therefore, this matching approach can not find any match candidate with a 2-edit distance of typographic errors. Since errors could occur at any position in a term, low match rates resolved by phonetic-encoded methods are expected as these two methods can not handle errors that change the first character of a term [49].

To draw the concrete conclusion on which matching technique should be used for matching street names, we first remove Error V since this error could be handled differently and summarize matching rates, runtime, and matching efficiency in Table 2.5. We first look at the match rate and runtime, as these two metrics could profoundly impact the final output match rate and the query latency for a geocoding process. As can be observed, these similarity based approaches, especially Jaro and Jaro-Winkler, have relatively higher probabilities to find a correct candidate than

Table 2.5: Overall street name match performance of each matching technique

	Match Rate (%)	Runtime (ms)	Match Efficiency
Similarity-based approach (simi > =0.5)			
Jaro	99.89	496.59	0.7816
Jaro-Winkler	99.94	682.42	0.7969
Edit Similarity	98.84	768.88	0.8011
2-grams Dice Coeff	94.25	652.86	0.6731
3-grams Dice Coeff	83.25	637.43	0.5259
Filter-based approach			
Soundex	69.50	–	0.0008
Double-Metaphone	64.83	–	0.0006
Edit Dist T1	64.13	453.81	0.00003
Edit Dist T2	88.31	608.54	0.0001
Pad 2-grams T1	98.37	123.08	0.3838
Pad 2-grams T2	98.23	91.41	0.1265
Pad 3-grams T1	98.34	39.12	0.1512
Pad 3-grams T2	98.12	24.63	0.0319

the remaining approaches. However, such high match rates also require more time to complete the needed comparisons, as indicated by the empirical runtime measurements. Because the phonetic-encoded strings (i.e., Soundex and Double-Metaphone) can be calculated and indexed beforehand, we did not explicitly measure their runtime. Among matching techniques, Pad q-gram based approaches have a relatively small runtime while maintaining comparable match rates to Jaro-based approaches and Edit Similarity. With an increase in the threshold of the number of shared grams to be a matching candidate (denoted as T1 or T2) and granularity of word grams (i.e., q of 2 or 3), the match rates of Pad q-gram based approaches slightly reduces and the speed to complete comparisons increases. In terms of matching efficiency, Edit Similarity has the best performance among these measured techniques. The two approaches from the Jaro family perform closely, with Jaro-Winkler having a slightly higher MRR than Jaro. For filter-based approaches, Pad q-gram based approaches produce significantly more candidates than other techniques due to the relaxed rule to consider matching candidates. When the threshold and the size of the word gram are small, it is relatively easy for Pad q-gram based methods to find shared n-gram(s), even for totally dif-

ferent strings. For example, two strings: "East" and "Earl" could be considered a match because these two words share one of 2-gram: "ea". Since the produced matching candidates are primarily used as conditions of query strings, the importance of this metric depends on the query languages and data warehouse used.

Finally, we explore the performance of hybrid methods under a different number of returned candidates. Such a measurement can be treated as the method for selecting matching techniques, with considering the geocoding matching process performance, as we tend to select a matching techniques that have high match rates while returning a small number of candidates. Based on Table 2.5, we choose Pad 3-grams T2 as the filter function owing to its relatively high match rate and short runtime in comparison to other filtering functions, and choose similarity-based methods that have a higher match rate than such a filter function: (1) Jaro, (2) Jaro-Winkler, and (3) Edit Similarity to compose hybrid method variations. We then feed the same testing dataset used for Table 2.5 into each hybrid method and compare the overall performance of each hybrid method variation with the retrieval of the at-most first 30, 50, 100, and all candidates. Table 2.6 lists the overall performance of each hybrid method variation. Jaro + Filter 10 and Jaro + Filter All indicate hybrid methods that are composed of the Jaro similarity-based method and the Pad 3-grams T2 filter based method, returning only up to the first 10 and all candidates, respectively. As can be observed, when retrieving all candidates, the match rates of the three evaluated hybrid methods are higher than the selected filter method (i.e., Pad 3-grams T2) and are lower than the corresponding similarity methods: Jaro, Jaro-Winkler, and Edit Similarity listed in Table 2.5. This is expected as a similarity-based method that performs similarity calculations upon candidates selected by a filter-based method; thereby, filter methods and similarity-based methods control the lower bound and upper bound of the match rate for the composed hybrid method, respectively. For hybrid methods that utilize the same similarity based method, their match rates increased as the number of retrieved candidates increased. This is because similarity based methods did not yield high ground-truth data rankings, resulting that correct candidates being excluded from the retrieved candidates (e.g., the first 30 candidates). Overall, we observed that the hybrid method that is composed by Edit

Similarity and Pad 3-grams T2 has the best matching performance, as such a hybrid method yields the highest match rates at every level of retrieved candidate quantity.

Table 2.6: Overall performance of matching street base name under hybrid matching method variations

	Match Rate (%)	Match Efficiency
Hybrid-based approach		
Jaro + Filter All	99.66	0.7842
Jaro + Filter 100	94.98	0.7841
Jaro + Filter 50	93.61	0.7840
Jaro + Filter 30	92.42	0.7832
Jaro-Winkler + Filter All	99.66	0.7969
Jaro-Winkler + Filter 100	95.98	0.7969
Jaro-Winkler + Filter 50	94.77	0.7968
Jaro-Winkler + Filter 30	93.63	0.7967
Edit Similarity + + Filter All	99.67	0.8010
Edit Similarity + + Filter 100	96.35	0.7999
Edit Similarity + + Filter 50	95.09	0.8005
Edit Similarity + + Filter 30	93.92	0.7992

City: We analyze the performance of matching techniques for city names similar to the way of analyzing matching techniques for Street base names. Table 2.7 summarizes the match rates of each matching technique facing different errors of city names. As can be seen, most matching techniques except certain Pad q-gram based methods have relatively lower match rates on Error V - "First Character Abbreviations", because this error tremendously decreases the length and characters of the original city name string (e.g., Los Angeles will be converted to LA.) Different from other matching techniques, Pad 2-grams T1 and Pad 3-grams T1 consider two records as a match candidate as long as it has a common character with the input string. Therefore, in the case of Error V, these two matching techniques can still find the correct matching candidates, since the first character of the city name abbreviation and its corresponding reference data are the same. Such a relaxing rule (i.e., threshold of 1) also explains the superior performance in terms of match

Table 2.7: Match rate of matching techniques under different city name error categories

Match method \ Error category	I	II	III	IV	V	VI	VII	VIII
Similarity-based Approach (simi >= 0.5)								
Jaro	96.89	96.89	94.41	97.40	94.09	95.01	96.35	83.44
Jaro-Winkler	96.89	96.89	94.41	97.40	94.30	95.08	96.44	83.83
Edit Similarity	96.90	96.89	94.45	97.40	0	94.30	90.02	86.04
2-grams Dice Coeff.	96.34	82.60	94.45	97.40	0	92.98	84.11	97.39
3-grams Dice Coeff.	85.28	50.91	94.41	97.40	0	91.40	80.90	85.28
Filter-based Approach								
Soundex	66.95	43.29	50.69	96.02	0.07	68.73	66.06	36.24
Double-Metaphone	60.84	36.32	54.76	92.73	0.02	68.47	61.81	35.33
Edit Dist. T1	96.90	0	0	97.40	0	0.06	28.39	0
Edit Dist. T2	96.89	96.90	46.99	97.40	0	15.56	29.34	0.39
Pad 2-grams T1	100	100	100	100	100	100	100	99.98
Pad 2-grams T2	100	99.32	100	100	11.14	99.74	100	99.74
Pad 3-grams T1	100	100	100	100	100	100	100	99.89
Pad 3-grams T2	100	98.57	100	100	8.10	99.71	100	99.61

* Error category: (I) One edit distance typographical error; (II) Two edit distance typographical error; (III) Additional directional; (IV) Additional white-space; (V) First character abbreviation; (VI) Partial abbreviation; (VII) Semantic error; (VIII) Direction omission.

rates across different error types.

Table 2.8 summarizes the match rate, match efficiency, and runtime for each matching method after removing Error V. In terms of the match rate, Jaro, Jaro-Winkler, and Edit Similarity outperform the remaining similarity-based methods and most of filter-based approaches, which is similar to the pattern found in matching street names. The only exception is that Pad 2-grams T1 and Pad 3-grams T1 have a higher match rate than these three similarity methods because of the relaxed rule of Pad q-gram based method for considering a match, which is also observed in Table . In terms of runtime, as long as a matching method has a similarity calculation involved, the runtime is slow, which is consistent with what has been detected in matching street base names evaluations. Likewise, the patterns of match efficiency in matching city names and street base names are similar. Among similarity-based approaches, Edit similarity still has the highest match efficiency; 2-grams Dice Coeff and 3-grams Dice Coeff have a better performance in ranking the ground-truth city

names than street base names. For the filter-based approaches, the number of matching candidates generated by Pad q-gram based approaches are larger than the number of candidates produced by the remaining filter based approaches. For example, the Pad 2-grams T1 method produces over 3,000 times the candidates than Edit Dist T2. Such a large number of candidates can cause issues for constructing query strings. Therefore, a hybrid method that combines similarity-based and filter-based methods is worthy of investigation.

Table 2.8: Overall performance of matching city name under each matching technique

	Match Rate (%)	Runtime (ms)	Match Efficiency
Similarity-based approach (simi > =0.5)			
Jaro	95.91	20.86	0.8141
Jaro-Winkler	95.92	28.03	0.8180
Edit Similarity	95.60	43.36	0.8468
2-grams Dice Coeff.	90.87	28.91	0.7573
3-grams Dice Coeff.	76.95	30.39	0.6326
Filter-based approach			
Soundex	48.19	–	0.0008
Double-Metaphone	44.99	–	0.0006
Edit Dist. T1	28.97	34.53	0.00003
Edit Dist. T2	63.61	34.21	0.0001
Pad 2-grams T1	99.99	3.84	0.3838
Pad 2-grams T2	86.67	3.61	0.1265
Pad 3-grams T1	99.97	1.83	0.1512
Pad 3-grams T2	86.01	1.15	0.0319

Based on Table 2.8, we select Pad 3-grams T1 as the filter function and pair with Jaro, Edit Similarity and Jaro-Winkler methods to compose a hybrid method, since these three similarity-based methods have higher match rates than remaining methods. We summarize the overall performance in Table 2.9 for different method variations. Similar to the way we construct a hybrid method for street base names, variations of hybrid methods come form the selection of similarity score methods (i.e., Jaro or Edit Similarity or Jaro-Winkler) and number of returned candidates. In this table, Jaro + Filter All and Jaro + Filter 30 denote hybrid methods are compose of Jaro and Pad 2-grams

T1 and return all candidates and up to first 30 candidates, respectively. As can be observed, hybrid methods that return all candidates have the highest match rates, and the match rate of each hybrid method variation decreases with the decreasing in the number of returning candidates. This is similar to the pattern observed for matching street base names in Table 2.6, as a high match rate requires a hybrid method gives the correct candidate a relatively high ranking position. Overall, Edit Similarity + Filter 100 can be considered the best matching technique for city names as it produces the highest matching rate while keep number of returned candidates to be relatively small.

Table 2.9: Overall performance of matching city name under hybrid matching method variations

	Match Rate (%)	Match Efficiency
Hybrid-based approach		
Jaro + Filter All	99.53	0.8499
Jaro + Filter 100	98.18	0.8497
Jaro + Filter 50	97.10	0.8494
Jaro + Filter 30	96.02	0.8491
Jaro-Winkler + Filter All	99.56	0.8514
Jaro-Winkler + Filter 100	94.41	0.8512
Jaro-Winkler + Filter 50	93.60	0.8511
Jaro-Winkler + Filter 30	93.04	0.8509
Edit Similarity + Filter All	99.23	0.8819
Edit Similarity + Filter 100	98.97	0.8825
Edit Similarity + Filter 50	98.59	0.8823
Edit Similarity + Filter 30	98.14	0.8832

Street-level descriptions: As described in Section 2.4.1, we seek matching candidates only if address parsing results are coarse, namely, an address parser is not able to extract fine-grained BKVs. Therefore, we only evaluate how well classical token-based approaches can be applied to handle low-quality street-level descriptions since the length of a street-level description is relatively longer than a street base name or a city name only. To conduct evaluations, we index the entire Navteq2016 reference dataset into Elasticsearch so that the term frequency and ranking score are the same as real production geocoding scenarios. Because we aim to evaluate whether or not it

is feasible to seek matching candidates utilizing street-level descriptions without further parsing, we directly quantify their overall performance of each matching method variation for short and long street-level descriptions, as shown in Table 2.10 and Table 2.11. This helps us getting a notion of which method is the most efficient approach for matching street-level descriptions and is competitive to methods that use street base name and city name fields to conduct matching.

Table 2.10: Overall match performance of each matching technique for short street-level descriptions

	Match Rate (%)	Runtime (ms)	Match Efficiency
TF-IDF	52.59	381.71	0.2089
BM25	52.40	234.73	0.2171
2-grams TF-IDF	75.29	1041.45	0.3314
2-grams BM25	75.32	279.48	0.3328
3-grams TF-IDF	71.17	298.93	0.3195
3-grams BM25	71.76	211.09	0.3242
Pad TF-IDF	52.72	361.24	0.2594
Pad BM25	52.42	254.08	0.2722
Pad 2-grams TF-IDF	75.29	980.79	0.3327
Pad 2-grams BM25	75.32	267.99	0.3341
Pad 3-grams TF-IDF	71.20	294.33	0.3330
Pad 3-grams BM25	71.77	211.10	0.2485
Fuzzy Pad TF-IDF	71.56	455.75	0.3374
Fuzzy Pad BM25	72.25	349.57	0.3681
Fuzzy Pad 2-grams TF-IDF	79.98	988.02	0.3883
Fuzzy Pad 2-grams BM25	75.32	272.50	0.3864
Fuzzy Pad 3-grams TF-IDF	69.50	1253.13	0.1983
Fuzzy Pad 3-grams BM25	68.11	553.71	0.1895

Match rates for short and long street-level descriptions show a similar pattern. In terms of token resolutions, the individual term resolution has the lowest match rate using either the TF-IDF or the BM25 scoring function, compared to 2-grams and 3-grams, due to the fact that any character change would lead to mismatches at the level of individual terms. The 2-grams resolution is found to have a higher match rate than the 3-grams resolution; this is because some street descriptions

Table 2.11: Overall match performance of each matching technique for long street-level descriptions

	Match Rate (%)	Runtime (ms)	Match Efficiency
TF-IDF	59.85	441.35	0.2571
BM25	59.44	237.68	0.2697
2-grams TF-IDF	79.96	1046.13	0.4145
2-grams BM25	79.99	279.81	0.4161
3-grams TF-IDF	72.03	307.68	0.2399
3-grams BM25	72.74	219.05	0.2458
Pad TF-IDF	60.03	421.73	0.4320
Pad BM25	59.64	233.79	0.4564
Pad 2-grams TF-IDF	79.98	1004.37	0.3227
Pad 2-grams BM25	80.04	271.43	0.3341
Pad 3-grams TF-IDF	72.02	301.17	0.3363
Pad 3-grams BM25	72.75	212.27	0.3416
Fuzzy Pad TF-IDF	71.56	510.92	0.3373
Fuzzy Pad BM25	72.25	253.91	0.3341
Fuzzy Pad 2-grams TF-IDF	79.98	1023.03	0.3327
Fuzzy Pad 2-grams BM25	80.04	279.20	0.3341
Fuzzy Pad 3-grams TF-IDF	69.51	1221.34	0.2854
Fuzzy Pad 3-grams BM25	68.11	565.40	0.2783

with a relatively short string length (i.e., ordinal number street names or abbreviation of street directions) may not be split properly by a 3-grams segmentation. This also explains why the padded version of 2-grams and 3-grams terms have slightly higher match rates than 2-grams and 3-grams terms without padding; namely, with the help of additional padding characters, strings will be enough to be split into different chunks. We employ the default fuzzy matching strategy (i.e., up to a 2-edit distance) from Elasticsearch to see if it could help in handling errors and variations to improve the match rate. However, the benefit of a fuzzy query is only observed for the individual term resolution, since 2-grams and 3-grams can already handle the certain string mismatch scenarios. In terms of similarity scoring methods, BM25 outperforms TF-IDF in most cases except for the individual term resolution, and BM25 also shows a shorter runtime compared to the TF-IDF method.

Among all evaluated token-based methods, Pad 2-grams BM25 has the highest match rate while

maintaining a relatively short runtime and a relatively high match efficiency (i.e., MRR). However, match rates of street-level descriptions resolved by token-based methods are not comparable to match rates for these evaluated fine-grained BKVs. Thus, using street-level descriptions to retrieve candidates should only be considered when fine-grained parsing results (i.e., address component level) are not available.

2.4.3 Matching Logic Evaluation

The ultimate step to determining the performance of the entire matching process is to assemble a query string that pairs each individual blocking field with the corresponding matching techniques; we term such a process as *matching logic*. Although we obtained the most suitable matching technique through assessments in the Section 2.4.2, the performance of a query string that queries applicable address components for an input address description remains uncertain and requires further evaluations due to the following reasons: (1) Because of the logic operations, the summation of the best matching techniques on each BKV may not always lead to the best overall matching performance. For instance, once a matching approach fails to return the correct city name with respect to the input, a query string that requires strict matches to both city and street names will have zero candidates to be returned. Moreover, a query string has to deal with situations such as when a blocking field is omitted from the input queries. (2) A matching logic may return a huge number of candidates as field string matching techniques attempt to include many matching candidates for each BKV. For example, Pad q-gram based methods are expected to return zip codes outside the target administrative areas. It is possible that the returned candidates may have the same descriptions but are located at different administrative areas, especially for address records that have common descriptions such as "main street". (3) When considering assembling a query string, more information can be leveraged to potentially speed up the query execution. For example, using the provided state name or the derived state name from a zip code, street names that are located outside the target state can be filtered out, significantly reducing the needed comparisons between the input street name and reference datasets. Given these uncertainties, our evaluations attempt to answer which matching logic is robust to low-quality input, yielding the best matching candidate quality

and the lowest query latency.

2.4.3.1 Data and Evaluation Metrics

Data

Datasets used in this evaluation include input dataset and address reference dataset. For reference datasets which is used for querying, we index the whole Navteq2016 datasets into Elasticsearch so that the dimension of search space for queries can be the same as that in real scenarios. We use both 1-error and 2-error testing input dataset generated by 2.3 as the input to evaluate the matching performance facing input with different degrees of errors.

Differ from errors introduced in evaluations for individual fields, we include errors of *omission* (e.g., a city name is omitted from the input query) and the unpaired zip codes and State names, which commonly exist in real scenarios. To simulate geocoding input with various quality [50], we generate two testing datasets by randomly injecting one and two errors into address records, respectively.

Evaluation metrics

We choose match rate and runtime to quantify the performance of each matching logic variation. We consider that match rates as the most important metric as if no correct candidates can be retrieved at this stage, the entire ranking process will become meaningless. As for runtime, the query latency of a matching process could contribute to the time consumption for the entire geocoding process, which becomes critical when facing large amounts of address records [19].

2.4.3.2 Evaluation Procedures and Results

To assemble query strings, we consider each address component based on their contributions to finding likely match candidates. Street base name is the most important identifier when searching for a candidate, even in the case that the input house number does not exist in the reference datasets, we are still able to find a nearby candidate on the street [27, 63]. Thus, every constructed query must hit against the field of street name. Since the field of city and zip codes could help to locate the candidates in the correct administrative area, reducing the search space for street base names,

we consider that likely-matching candidates must match the input street base name and at least match one from the set of input city names, state names and zip codes. To this end, we formalize three matching logic variations as follows.

- Variation 1: Street base name $\cap$ {City, State, Zip code}
- Variation 2: Street base name $\cap$ {State, Zip code}
- Variation 3: Street base name $\cap$ {City, State}
- Variation 4: Street base name $\cap$ Zip code
- Variation 5: Street base name $\cap$ City

Each matching logic is expected to generate a query string which hits against these fields listed above. Since errors occur in a zip code could lead to this zip code points to a new state, if both zip code and state exist in input queries, the matching logic would check if zip code and state are paired or not. If not, the matching logic adds the state that corresponds to the zip code. To deal with the error of "First letter abbreviation" occurs in the field of city names, we build a dictionary that can help the query strings to convert the abbreviation of city first letters to original city names. Variation 1 leverages all possible administrative area information provided by input queries, namely, this variation attempts to match city, state, and zip code as long as these address components are provided. Compared to Variation 1, Variation 2 and Variation 3 only leverage up to 2 different administrative levels. Variation 4 and variation 5 attempt to an administrative area that is smaller than the state. We exclude the matching logic that only utilizes state names as the only field to query against because it is more likely that one can find multiple locations with the same street name within a state such as "main street". These variations represent different degrees of matching demands (i.e., matches to different numbers of blocking fields) to identify output candidates as these query strings are attempting to leverage spatial constraints among matching fields to reduce search space. However, the pros and cons of such variations remain uncertain and are worth exploring. For example, the difference between the first two matching variations is the requirement

of matching city names. On one hand, the additional query criterion of city name could reduce search space for matching candidates. On the other hand, such a query criterion of city name may introduce more uncertainty to candidate retrieval, since there is no guarantee that we can always find the correct city name to build query strings when there are errors in the input city names.

To conduct an evaluation, we first select the best matching techniques based on the evaluations in Section 2.4.2. Namely, we use Pad 2-grams T2 for zip codes, and use for Edit Similarity + Pad 3-grams T2 and Edit Similarity + Pad 3-grams T1 for street base name and city. In terms of constructing a query string, we select first 30 and 100 candidates that are ranked by string similarity values for fields of street base name and city name as matching conditions for these two fields. To limit the number of candidates being returned, we rank candidates by similarity values between input and reference street base name based on Equation 2.1 and only return the first 500 candidates. We then feed each logic variation by 1-error and 2-error testing input datasets and use match rates and runtime to quantify its matching quality. Table 2.12 lists the performance of the

Table 2.12: Performance of each matching logic variation facing 1-error and 2-error testing input datasets

(a): 1-error dataset

	Match Rate (%)	Query Latency (ms)
Variation 1	92.57	47.12
Variation 2	**93.30**	61.66
Variation 3	89.88	63.50
Variation 4	88.54	73.04
Variation 5	66.14	98.52

(b): 2-error dataset

	Match Rate (%)	Query Latency (ms)
Variation 1	87.83	60.39
Variation 2	**89.00**	58.87
Variation 3	83.96	73.82
Variation 4	82.85	89.86
Variation 5	63.42	103.52

above matching logic variations facing one and two degrees of errors. Two tables show a similar pattern, that is, Variation 1 and Variation 2 have higher match rates and lower query latency than the remaining matching logic variations, the performance of matching logic decrease with less spatial constraints, and Variation 5 have worse matching performance in terms of these two metrics. Such a pattern can be explained by how errors occur in a field and underlying matching techniques we used. For example, the unmatched case distribution of Variation 3 under 1 degree of error scenario indicates that this variation is more likely to fail to include the ground-truth data in its output list once errors touch the field of city and state, as the count of errors that omit the fields of state or city and the errors that change city or state point to a new place is larger than the count of each remained error. This also helps explain why Variation 5 produces a low match rate. Compared to Variation 5, Variation 4 can still yield relatively high match rates, because the selected matching technique for zip code, Pad 2-grams T2, is robust to zip code errors as shown in Table 2.3. Using input with 1 degree of error as an example, after excluding the zip code omission error, roughly 82% input containing zip code related errors have been resolved to include ground-truth data in their output list. When facing input with 1 degree and 2 degrees of errors, Variation 1 and Variation 2 have similarity performance, which is confirmed with their unmatched case distributions. Under these two input dataset scenarios, the error of "number to word representation" dominates in error distributions, as such an error is found to handle not very well by the evaluated fuzzy string methods in Section 2.4.2. Under the 2 degrees of errors input scenario, Error - "zip code and state unpaired" also dominates the unmatched case distribution. Based on the designed implementation, this error causes the matching logic to search address candidates in multiple states (i.e., the provided state and the state that derives from an input zip code). If the provided street name exists in multiple states, the matching logic is observed to return a candidate that has the same description as the input but is located in another state, thereby reducing the match rate. This is because a query string is designed to conduct a rudimentary search process to include candidates as much as possible, and all candidates are ranked by their textual similarity. The ground-truth candidate can be easily filtered out, if one sets up a small number of candidates to be returned. This observation suggests

that the way to build up a query string can be adaptive to the frequency of a query field (e.g., street or city names). Namely, if a provided street name commonly exists in multiple places (i.e., an address component frequency is high across the entire address reference dataset), we would construct a query string with more strict spatial constraints such as limited number of zip codes. Variation 2 has better match rates than Variation 1. This can be explained by the additional city field query conditions made by Variation 1. Once such a query condition fails, no candidate can be retrieved. In terms of query latency, Variation 1 only shows its advantage of using city name as an additional query condition under 1 degree of error scenario. Overall, the first two matching logic variations are expected to be robust to low-quality input and Variation 2 is the most robust match logic, given the matching techniques used.

2.5 Benchmark for Geocoding Ranking Process

In this benchmark, a *Ranking* process is defined as a process used to assign each candidate retrieved by a matching process that scores and ranks these candidates in a quantitative manner. Thus, such a process determines how each candidate is ranked in the final output list. It is possible that even if the previous matching approach has retrieved the correct candidate from an input, this candidate may not appear in the final output list because the ranking process failed to give it a ranking position higher than the total number of candidates being returned. Typically, a candidate who appears in the first position of the output list is considered the best. Given the context of the geocoding text retrieval process, we consider the best candidate to either have high textual similarity or be fully identical with respect to the input. Therefore, the goal of the ranking process is to assign each candidate a score with the expectation that the best candidate can be ranked in the first position.

2.5.1 Evaluated Ranking Approaches

We define a ranking process inside a geocoding workflow as a rank that retrieves address candidates in a quantitative manner based on textual similarity with respect to the input address description. In geocoding literature [43, 20], a ranking process is often considered as being tied to

two categories of matching mechanisms: deterministic-based and probabilistic-based approaches. A deterministic-based approach empirically assigns these candidate scores using a defined penalty scheme and selects a threshold used to differentiate between matching statuses (i.e., matched vs. unmatched). In contrast, a probabilistic-based approach draws scores based on the importance of an address component in determining the matching status. In the case of probabilistic-based approaches, the quantification for a geocode candidate could come from term frequency, string similarity, or both. To this end, we formalize the baselines covering both categories of ranking approaches. As we consider looking up address candidates as a text retrieval process, we also include recommendation-based ranking approaches from the domain of information retrieval (IR) to evaluate its feasibility for ranking address descriptions. We evaluate each approach, as detailed in the following paragraphs.

2.5.1.1 Per-attribute Score Ranking

As the name applies, **Per-attribute Score Ranking** scores a reference candidate on the basis of each individual address component. The score for each component depends on two factors. First is an empirical-based weight, depending on the importance of an address element in determining a match status. For example, the field of street base name, considered to be more important to identifying a correct candidate, has a higher weight than the state field. Second is the quantification of similarities between input queries and reference candidates based on the nature of the address components. For example, the field of street base name is typically measured with fuzzy string techniques, and a state name can be quickly compared in a gazetteer lookup manner. Table 2.14 summarizes the weight and similarity quantification used for each address component. For the weights of the address components, we directly used the list of weights utilized by Texas A&M GeoServices, given its performance in a variety of studies [18, 63, 28]. To quantify similarity, we employ the edit distance-based similarity method (Equation 2.1) to calculate the similarity between the input and reference data. For zip codes, we quantify the similarity of two zip codes as follows.

$$\text{Similarity} = 1 - 0.2 \times N \tag{2.3}$$

where N is the number of digits that is not the same as that in the same position of the reference zip code. Since house numbers in a street block has maximum numeric difference of 100 according to USPS TIGER[5], we consider two house numbers are completely different if their numeric difference is larger than 100. We quantify their similarity in Equation 2.4.

$$\text{Similarity} = 1 - \frac{|HN_{ref} - HN_{inp}|}{100} \tag{2.4}$$

Where HN_{ref} and HN_{inp} represent house number in the reference data and in the input, respectively. For the remaining address components, namely, street pre- and post-direction, road type, and state, the similarity is quantified in a Boolean manner, namely, the similarity is 0 if two compared fields are not identical, and the similarity is 1 if they are the same.

Table 2.14: Weight and similarity quantification of each address component for per-attribute score ranking

Address Component	Weight	Similarity Quantification
House number	20	Equation 2.4
Pre-Direction	7	Boolean
Street base name	45	Equation 2.1
Road type	10	Boolean
Post-Direction	4	Boolean
City	17	Equation 2.1
State	1	Boolean
Zip code	45	Equation 2.3

2.5.1.2 Term Frequency-based Ranking

Originally, term frequency-based methods in the geocoding domain attempted to weigh the importance of an attribute based on the number of appearances in the reference datasets. Namely, if a term appears too often across the entire reference datasets, such a term is considered to be less

[5] `https://www2.census.gov/geo/pdfs/maps-data/data/tiger/tgrshp2017/TGRSHP2017_TechDoc_Ch4.pdf`

valuable for finding a true matching candidate, therefore receiving a lower weight. In the information retrieval domain, a classic ranking method known as *Term Frequency - Inverse Document Frequency* (TF-IDF) uses a similar principle to calculate the similarity value between an input query and a document. Specifically, the more frequently a term appears in a document, the more important this term is, whereas, if a term exists in multiple documents in a collection, such a term is less important. Given that TF-IDF has been widely used for document relevance calculations in various information retrieval systems such as search engines, we adopt this approach to quantify similarity between geocoding input queries and address descriptions in the reference dataset. We limit our use of this approach to fields of street and city names. As remaining address components (such as street directions), after normalization, have fixed term numbers and can be easily compared in a Boolean manner, the same Boolean comparison method is used in Section 2.5.1.1. In total, we evaluate the performance of two variations of TF-IDF-based methods for relevance score calculations, which are built upon the default scoring function of Elasticsearch, given the wide usage of Elasticsearch in existing geocoding systems[60]. The first method is to calculate scores purely from a term frequency perspective. Specifically, IDF for a term is calculated as: $IDF = \ln \frac{N+1}{n+1} + 1$, where N is the total number of documents that have this queried field and n is the number of documents that contain queried terms. We add 1 to the IDF to prevent it from becoming zero. The weight of this term is calculated as

$$W = \sqrt{tf} \times (\ln \frac{N+1}{n+1} + 1) \times \frac{1}{\sqrt{dl}} \tag{2.5}$$

where tf denotes the frequency of a query term and dl denotes the length of fields for retrieved records. We term this method as **Classic TF-IDF**.

The other approach aims to provide more tolerance for mismatching that occurs between input description and address reference data. By default, Elasticsearch conducts fuzzy queries within a tolerance of up to two edit distances; namely, these scores are calculated for candidates that have less or equal to two edit distances with respect to the input. Based on the evaluation results

from Section 2.4.2, we learned that such edit distance threshold-based methods could not handle certain categories of input errors very well, such as input contains partial abbreviations. Therefore, we embrace the Soft TF-IDF method, using string similarity as the second weight function upon TF-IDF [58] as follows.

$$W = \sqrt{tf} \times (\ln \frac{N+1}{n+1} + 1) \times \frac{1}{\sqrt{dl}} \times Sim \tag{2.6}$$

where Sim is edit distance based similarity, since such similarity values have been calculated during the matching phase, we can save certain computational expense. We denotes this method as **Similarity TF-IDF**. As for similarity quantification, we employ the same method from Table 2.14 except that we use Equation 2.7 for zip codes to better differentiate the mismatch at different digit position P.

$$\text{Similarity} = 1 - (5 - P)/15 \tag{2.7}$$

2.5.1.3 Classification-based Ranking

The classification-based ranking employs a statistical-based classifier to predict a matching status for each candidate, and all output candidates are ranked according to their predicted status. This approach is widely used in entity matching scenarios [61, 48] and scenarios in which only the best geocoded data needs to be generated rather than to provide a list of ranked candidates [25]. To rank candidates in a meaningful way, we define two sets of matching status labels as follows.

2-Label:

1. Match: a output geocode represents the exact location of the building that has the same address as the input query.

2. Unmatch: a output geocode can not represent the exact location of the building that has the same address as the input query.

3-Label:

- Point match: a output geocode represents the exact location of the building that has the same address as the input query.

- Street-level match: a output geocode can not pinpoint to the exact location of the input address but could match other locations on the same street as the input. Such output geocoded data could be considered as the best alternative output in the case that the exact location of the input address is not available.

- Unmatch: a output geocoding candidate has a spatial accuracy lower than a street-level match.

The first set of labels simply treats a geocoded process as a match/un-match task for all candidates. In contrast, the second set of labels, which are based on the NAACCR geocode quality [28], are able to differentiate between candidates with a street match level or lower than street match level. To conduct the classification, each address component needs to be compared and quantified by using string similarity. As for similarity quantification methods, we use the same quantification methods listed in Table 2.14 for each address component. In terms of the classifier model, we chose the Random Forest classifier [71], owing to its proven and robust over-fitting in a previous geocoding classification task [25].

Since a training process is required to build such a classifier, we first train a classification model with geocoding input queries and their matching candidates. Specifically, we sample input data from the Navteq 2016 reference dataset, excluded the address records in the 1-error and 2-error testing input dataset, and fed these data into the match process defined in Section 2.4.3 to obtain a set of matching candidates for each input query. We then iterate and generate a set of similarity values for each address component for every matching candidate. Since we know the corresponding ground-truth data for every input data, we add the defined matching by comparing address descriptions. We randomly select five records from every set of retrieved matching candidates and ensured that these sampled records included labels of point match and street-level match and contained the label of unmatchable when applicable. To this end, we process 100,000 input queries.

For each input, there are five corresponding tuples, which are composed of seven similarity values (i.e., one similarity value per address component) and one match label. To build a robust random forest classifier, we fine-tune hyper-parameters of a random forest model in an exhaustive search manner, using scikit-learn library[6]. Specifically, we iterate each combination of hyper-parameters drawn from the candidate parameter pool listed in Table 2.15 using three cross-validations to select parameters that lead to the best F1 score. To this end, we obtain two sets of best parameters (listed in Table 2.15) that produce the best weighted F1 score of 0.9829 for the 2-labels scenario and 0.8875 for the 3-labels scenario, respectively. Next, we build a random forest classifier using

Table 2.15: The candidate parameter search space and the best parameters for Random Forest Classifier

Hyper-parameter	Description	Candidate Parameter	2-Label Best Parameter	3-Label Best Parameter
Number of trees	The number of trees in the forest	100, 300, 500, 1000, 1200	100	300
Max tree depth	The maximum depth of the tree	5, 8, 15, 25, 30	15	15
Min sample split	The minimum number of samples required to split an internal node	2, 5, 10, 15, 100	25	25
Min sample leaf	The minimum number of samples required to be at a leaf node	1, 2, 5, 10	1	1
Criterion	The function to measure the quality of a split	gini, entropy	gini	gini

the best parameters listed in Table 2.15 and use all the training data to train the classifier. Finally, we feed the testing input data into the geocoding system and use retrieved candidates from the previous matching process as the input for the ranking process. Each retrieved candidate is used to generate a tuple of similarity values calculated from Table 2.14, and the classifier uses these similarity values to generate a matching label. In the case that multiple candidates that have the same matching label, we give a higher rank to candidates with similarity values.

2.5.1.4 Hybrid Ranking

By far, the evaluated term frequency-based and classification-based methods only considered the similarity between input and reference data from the perspective of string similarity. The limitations of these methods were observed in certain information retrieval tasks due to their inefficiency

[6] `https://scikit-learn.org/stable/`

in examining the semantic similarity between two compared fields [72]. Recently, dense vector-based encoding methods have shown promising results in capturing the semantic meaning of words in various NLP tasks [73, 74, 75]. In principle, a word is represented as a high dimension vector, where words with similar meanings are clustered [76]. Given the success of word embedding approaches, some works have already integrated these techniques into typical information retrieval techniques and achieved improved document ranking quality by creating a multi-stage retrieval workflow [77, 57]. Therefore, we adopt these approaches to see if they could help capture both semantic meanings and term similarities among the input queries and reference datasets in the context of geocoding. We choose to obtain vector representations for a word from Word2Vec (W2V) [76] and FastText [78], as these two models use two distinct levels of information—word-based information and n-grams-based sub-word information, respectively—to generate word vector representations. For W2V, we use the pre-trained model training on Google News[7] and the pre-trained model training with subword information on Wikipedia 2017[8]. For each word, we utilize a universal vector embedding library: Magnitude [79] to extract a 300-dimension vector. In cases where a field contained more than one word, such as street names or city names, we fixe the dimension of the vector for such a field by averaging the vector for each individual word using Equation 2.8, the same approach used by [80].

$$V_f = \frac{\sum_1^n V_n}{n} \tag{2.8}$$

where n is the number of words in a field, V_n represents the vector for the n-th word and V_f represents the vector for a field. It is important to note that finding a field that has the highest similarity related to the input requires iterating the entire dataset and sorting the cross product between the input vector and the vector of every reference data. To reduce the computational complexity, we only compare the input field with a subset of candidates retrieved from the previous matching process, which is a common strategy used by [68]. For a matching candidate, its similarity to the input is quantified by two portion, one is the word-frequency-based similarity, which we use the

[7] https://code.google.com/archive/p/word2vec/
[8] https://fasttext.cc/docs/en/english-vectors.html

value calculated by Similarity TF-IDF, denoted as $Simi_{TD-IDF}$. The other portion is the similarity value (denoted as $Simi_{vector}$) calculated by two dense vectors that represent input and reference data, respectively. To this end, the similarity for a matching candidates is calculated by

$$Simi_{candidates} = Simi_{TD-IDF} + Simi_{vector} \tag{2.9}$$

Depending on the pre-trained models used to generate word embeddings, we denote these two evaluated hybrid ranking methods as **Hybrid Ranking W2V** and **Hybrid Ranking FastText**, respectively.

2.5.2 Data and Evaluation Metrics

Data

Since a ranking process occurs after a matching process, we feed 1-error and 2-error testing input datasets into the system, all retrieved candidates by the matching process are used as the input to each ranking method.

Evaluation Metrics

Ideally, the best output geocode should be ranked in the first position, and the quality of each candidate decreases as the position increases. Thus, we evaluate the ranking efficiency by two metrics: (1) Correctness of the First Candidate (CFC), which is the ratio that the best candidate is the ground-truth candidate; (2) MRR, which indicates output ranking quality of a geocoding system by from the perspective that how does the best candidate (i.e., ground-truth data) rank in the produced results. Since we already know the ground-truth address record that corresponds to each input address description, we can easily find out the ranking position of the ground-truth data in output candidate lists and calculate these two metrics.

2.5.3 Evaluation Procedures and Results

Given that the performance of a matching process can impact the quality of a ranking process, we utilized matching techniques and matching logic that led to the best match rate, based on the evaluation from Section 2.4.3, to retrieve a set of candidates for 1-error and 2-error testing input

datasets and used these retrieved candidates as input for each ranking method. We consider that the candidate who ranks at the first position is the best candidate, comparing it to the ground-truth data to determine the actual ranking position of the ground-truth data. Table 2.16 summarizes the performance of each ranking method under one and two degrees of errors. As can be seen, ranking

Table 2.16: Performance of each ranking method facing 1-error and 2-error testing input datasets

(a): 1-error dataset

	Correctness of the First Candidate	MRR
Per-attribute Score Ranking	**0.9145**	0.9188
Classic TF-IDF	0.6408	0.6969
Simi TF-IDF	0.8725	0.8895
Random Forest 2-Label	0.9112	0.9160
Random Forest 3-Label	0.9039	0.9107
Hybrid Ranking W2V	0.8917	0.9041
Hybrid Ranking FastText	0.8946	0.9043

(b): 2-error dataset

	Correctness of the First Candidate	MRR
Per-attribute Score Ranking	**0.8634**	0.8690
Classic TF-IDF	0.5957	0.6522
Simi TF-IDF	0.8193	0.8386
Random Forest 2-Label	**0.8634**	0.8673
Random Forest 3-Label	0.8607	0.8650
Hybrid Ranking W2V	0.8418	0.8544
Hybrid Ranking FastText	0.8418	0.8543

performance shows a similar pattern under 1-error and 2-error testing input datasets: (1) Per-attribute Ranking had the highest CFC and the highest MRR among all other categories of ranking methods; (2) Random Forest 2-Label had the close performance in comparison to Per-attribute Ranking; and (3) Each ranking approach performed better on 1-error over 2-error testing input data. For term frequency-based ranking, Simi TF-IDF performed better than TF-IDF, which shows efficiency to utilize similarity as the secondary weights. For Hybird ranking based methods, Hybrid

ranking FastText had a slightly improvement in the CFC and MRR, compared to Hybrid ranking method which uses W2V to conduct word embedding under both 1-error and 2-error testing input datasets. This observation indicates FastText [78] can be helpful to capture sub-word information. For Random Forest-based ranking, Random Forest 2-Label performed better than Random Forest 3-Label because predicting two labels is a relatively easier task than predicting three labels, which is consistent with the prediction performance indicating by the weighted F1 scores. There are two directions to continue to improve the performance of the classifier-based ranking methods. One way is to explore various approaches to quantify similarities between input and reference address descriptions. The other direction is to fine-tune the hyper-parameter of the classification model further or even using different classifiers that have better generalization capabilities.

2.6 Benchmark for Geocoding Parsing Process

Parsing is the process of decomposing the entire input address string into multiple segments based on the standard address components, such as house number, street name, and postal code [20]. This task is considered a domain-specific NER task. The challenges of this task come from how people name a street, such as using a state name (e.g., Texas Ave.). It is difficult for an address parser to differentiate between a street name and a state name. Such a case becomes more complicated if the state name is missing from the input query. An immature address parser could assign a label of *State* to Texas.

In the geocoding workflow, the output from a parsing process is used to construct matching candidate query strings and calculate scores for each matching candidate. According to Figure 2.1 and the impacts of BKVs on matching performance assessed in section 2.4.2, the granularity of address parsing output (i.e., whether there is a completed address description) is crucial for the quality of matching output and matching latency. For example, the best matching and ranking performance, as evaluated in sections 2.4.3 and 2.5.1, requires an address parser to at least correctly identify the base street name, state name, and zip code. Otherwise, we must employ street-level description matching techniques to complete the geocoding process, which performs poorly with low-quality input. For a ranking process (section 2.5), all ranking approaches demand an address parser to

correctly recognize each address component to precisely calculate the ranking score. Therefore, the goal of the evaluations in this section is to discern the most efficient parsing techniques for satisfying the requirements of the matching and ranking processes based on our previous findings when facing (erroneous) input.

2.6.1 Evaluated Address Parsing Approaches

In this section, we describe two main categories of approaches to build an address parser: the statistical-based approach and the neural network-based approach.

2.6.1.1 Statistic-based Address Parsing

Statistics-based address parsers fall into the category of supervised learning. The features of input address descriptions are extracted first and fed into a statistical model to learn the likelihood of a label, followed by the previous label. Therefore, variation in statistics-based approaches comes from the choice of extracted features and statistical models. For the statistical model, we chose conditional random fields (CRF) over the Hidden Markov Model (HMM) to build an address parser because the former can conduct sequential tagging by overcoming the dependency assumption made by the HMM [81]. We compare the performance of address parsing that employs two different sets of features—syntactic features and syntactic features combined with domain-specific features—which have been used by [82]. Table 2.18 summarizes the features we used to train a CRF model.

2.6.1.2 Neural Network-based Address Parsing

A neural network-based parser is built upon neural network architectures and is trained by labeled address descriptions. The neural network-based approach is an end-to-end solution compared to statistical-based models, which heavily rely on feature engineering. It is more easily being applied to other address systems if training data is available.

[83] splits the entire neural network architecture into three modules, from input to output: (1) distributed input representations, (2) context encoder, and (3) tag decoder. According to an empirical study by [84], different choices of implementations or configurations of each module

Table 2.18: Features used to train a CRF model

(a): Syntactic features

Feature	Description
lower case word	word in the lower-case format
lower case word-1	word that is one word before from the current word
lower case word+1	word that is one word after from the current word
isDigitOnly	is word only contains digits
isAlphabetOnly	is word only contains alphabet letters
isAlphabetDigit	is word contain both alphabet letters and digits

(b): Domain-specific features

Feature	Description	Possible Value
isRoadType	is a road type in address reference datasets	True, False
isDirection	is a direction in address reference datasets	True, False
isState	is a state name in address reference datasets	True, False
isZipLike	is consist of five digits	True, False

compose a variety of neural network architectures, leading to different NER performance [84, 85]. Therefore, we investigat "popular" settings for these three modules of a neural network architecture based on previous works and made comparisons of their performance when facing low-quality address description input.

Distributed input representations. This module converts words into vectors of numeric values, which can be fed into neural network layers. Depending on the portion of a word that is encoded (e.g., the entire word, sub-word, or combination of the word and sub-word level), the input representations can be split into word-level, character-level, and hybrid-level representations [83]. For word-level representations, we select two pre-trained word-embedding models to encode each word in address descriptions: W2V [76], which is trained by Google News corpus [9], and ELMo [86]. These two models are selected because they generate two distinct word embeddings: context-independent and contextualized word representations, respectively. In terms of utilizing word representations from these two models, we use the extracted word vectors from W2V di-

[9] https://code.google.com/archive/p/word2vec/

rectly, as word vectors of W2V are generated based on tokens. By contrast, word vectors of ELMo are represented in the format of three layers of a language model, thereby requiring an additional step to transform the three-layer representation into a single layer [87]. We choose the *Fixed Average* and *Learned Weighted Average* methods to conduct this computation because they achieved the best performance of the NER task in CoNLL 2003 and a domain NER dataset, respectively [87]. We denote these two variations of ELMo as **ELMo fixed average** and **ELMo weighted average**. Given that character-level representations can be helpful in extracting morphological information, such as the prefix or suffix of a word [88], we evaluate whether additional distributed representations could improve the performance of address parsing. Specifically, we select convolutional neural network (CNN)-based [88] and LSTM-based [89] character representations, which are considered two major streams, to generate such representations [83] and appended them to W2V, denoted as **W2V CNN** and **W2V LSTM**, respectively. ELMo is already using a CNN layer on the character-level features to derive word representations; therefore, we did not concatenate the character representation of ELMo. We also compare the performance of each input representation variation to the pure word vector of W2V.

Context encoder. The context encoder, which is the main component inside the neural network architecture, aims to capture sequential dependencies using neural network structures, such as CNN and recurrent neural networks (RNN). In this benchmark, we select bi-directional long short-term memory (Bi-LSTM) neural networks for their state-of-the-art performance in sequential tagging tasks [56]. Thus, the variation of the context encoder in this work comes from the configuration of hyperparameters for neural networks. From [84], we list the evaluated "high-impact" hyperparameters and their candidate parameter pool as follows. The bold values are the default or best parameter used in [84].

- Optimizer: {Nadam, **adam**}
- Dropout: $\{0.05, 0.1, 0.25, \mathbf{0.5}\}$
- Gradient Normalization: $\{\mathbf{1}, 3, 5, 10\}$

Tag decoder. After the context dependencies inside the sequential data are encoded, a tag decoder can predict tags for each token in the input sequence, which is also known as a classifier. In the field of NER, two distinct tag decoders are widely used at the final stage in deep learning-based NER systems: (1) the Softmax-based decoder, which labels the input sequence in a multi-class classification manner, and (2) the CRF-based decoder, which assigns labels with considerations of entire sequences, maximizing the joint probabilities of tags for input sequences.

2.6.2 Data and Evaluation Metrics

Data

Data used in this benchmark includes training, development, and testing datasets, as the evaluated baselines include supervised-learning-based address parsers, which require a training process. We use the 2-error testing input dataset because such a dataset contains more chaos address descriptions. To build up the training dataset, we randomly sample 10 address descriptions from the unique zip code, city, and state pairs and ensure that the selected address descriptions do not exist in the testing dataset. In total, we have 435,133 address descriptions in the training dataset. When a development dataset is desired, we further split such a training dataset at the ratio of about 7 : 3. To this end, all three datasets (i.e., training, development, and testing datasets) are mutually exclusive. Meanwhile, regardless of the category of address parsing techniques (i.e., rule-based, statistical-based, and neural network-based) being evaluated, the used datasets for training, development (if applicable), and testing are the same.

Evaluation Metrics

Given the context of address parsing, we quantify the parsing performance by the standard NER evaluation metric: F1 score calculated by the precision and recall of every address component based on the USPS address standard. Such a measurement indicates a parsing algorithm's capability to recognize all address components correctly, which is required by the geocoded output candidate ranking performance (i.e., scoring based on each address component) and can help improve the matching performance. Since annotated labels (i.e., address components) are imbalanced in the testing dataset, we use a weighted F1 score to quantify the performance.

2.6.3 Evaluation Procedures and Results

2.6.3.1 Performance of the Statistical-based Address Parsing

To build a robust CRF model, we fine-tune regularization parameters of the CRF model using the randomized search and 3-fold cross-validation provided by the sklearn library. Specifically, we configure the searching space for the coefficient for L1 and L2 regularization of a CRF model using Equation 2.10.

$$\begin{cases} \text{L1 regularization coefficient} = \text{Exponential distribution with } \lambda \text{ of } 2 \\ \text{L2 regularization coefficient} = \text{Exponential distribution with } \lambda \text{ of } 20 \end{cases} \tag{2.10}$$

As for the parameter searching strategy, we limit the maximum iteration to be 50 and use the weighted F1 score as the metric to evaluate the performance of the cross-validated model. After identifying the best parameters for a CRF model, we construct a CRF model with the training algorithm of Gradient descent using L-BFGS method and maximum iteration of 100, and apply such a model to the 2-error testing data and quantify the performance by the defined metrics under the scenarios that input features with and without considering domain-specific features as shown in Table 2.18. We summarize the best estimator of and the corresponding parsing performance on the testing input data using different input features in Table 2.20. As can be seen, the CRF model with

Table 2.20: The performance of the fine-tuned CRF models with different input features

Input Feature	Best Parameter	F1 Score
Without domain-specific features	c1: 0.284602 c2: 0.003667	0.9903
With domain-specific features	c1: 0.210872 c2: 0.026878	0.9904

both syntactic and domain-specific features slightly improves the parsing performance, compared to the CRF model that only utilizes syntactic features. This can be explained by the effect of domain

features selection. Even without the usage of the selected domain-specific features, the CRF model still can fully recognize fields of state and zip codes reasonable (i.e., both fields have F1 score of 1), thereby making these two features - *isState* and *isZipLike* not useful. Likewise, the feature of *isRoadType* and *isDirection* may only provide limited information. This is because that the large amount of training samples we used has already provided enough information about the road type and street direction fields, given the limited spectrum of these two fields. The observations suggest that feature engineering is of importance to a statistical model-based address parsing.

2.6.3.2 Performance of the Neural Network-based Address Parsing

To select a set of parameter configurations of neural networks that leads to the best parsing performance while reducing the trails of hyperparameter combinations, we determine the best hyperparameter in a progressive manner, similar to the approach used by [85]. Specifically, we compare the performance of configuration variations drawing from the same configuration category (e.g., distributed input representation) and fixed the remaining configurations by using the default configuration setup defined in section 2.6.1.2 each time. In terms of neural network architecture module priority (i.e., distributed input representations, context encoder, and tag decoder), we first determine the configuration of the distributed input presentation and tag decoder that led to the best performance, given that these two configurations function as the input and output layer of a neural network. The setup variations inside the context encoder only impact the structure and utilization of a neural network. When conducting evaluations, we implement the neural networks based on [90, 87]. The hyperparameter of the neural network was initially set up by the default values defined in section 2.6.1.2. The number of recurrent units was 100. For the training process, we set up the epoch to be 25, with a batch size of 32. To diminish the randomness of the neural network training process, we took the average weighted F1 scores of two runs for each configuration variation. Table 2.21 summarizes the parsing performance for the 2-error testing input dataset. In terms of neural network modules, the distributed input representation module have the highest impact on parsing performance among all three modules, and the F1 score varied from 0.9911 to 0.9924. The selection of the optimizer also have a relatively large impact on parsing performance. These

Table 2.21: Address parsing performance of each neural network configuration variations

Module	Configuration	F1 Score
Distributed Input Representation	W2V	0.9911
	W2V + CNN	0.9917
	W2V + LSTM	0.9921
	ELMo Weight Average	**0.9924**
	ELMo Fixed Average	0.9911
Tag Decoder	CRF	**0.9924**
	Softmax	0.9922
Context Encoder	Optimizer - adam	**0.9924**
	Optimizer - ndam	0.9913
Context Encoder	Dropout - 0.05	0.9916
	Dropout - 0.1	0.9918
	Dropout - 0.25	0.9920
	Dropout - 0.5	**0.9924**
Context Encoder	Gradient Normalization - 1	**0.9924**
	Gradient Normalization - 3	0.9922
	Gradient Normalization - 5	0.9923
	Gradient Normalization - 10	0.9921

two observations were consistent with the finding in [84]. For distributed input representation, word embeddings obtained from the ELMo model with a weighted average strategy yields the best parsing performance. For the tag classifier, CRF outperform Softmax, given that the label of an address component has dependencies on adjacent labels [85, 91]. Inside the context encoder module, the configuration by default parameters used in [85] also yields the best performance in the address parsing task, which indicates that predicting address parsing is similar to other NER tasks.

2.7 Conclusions

Motivated by the demand to evaluate and improve the robustness of a geocoding system to erroneous input, we defined benchmarks and use them to assess the geocoding text retrieval process for low-quality input. Specifically, we presented an approach to synthesize low-quality geocoding input that matches human input patterns by randomly injecting errors and variations based on historic

geocoding transactions. Next, we defined three benchmarks target parsing, matching, and ranking process, respectively. Each benchmark includes evaluation datasets, evaluation procedures, and evaluation metrics. Finally, we used the defined benchmarks to evaluate various geocoding parsing, matching and ranking techniques and revealed a set of approaches that are robust when facing erroneous input queries. Evaluation results produced by the defined benchmarks also can be used as baselines to facilitate future geocoding evaluation and development works. The findings of each sub-task benchmark are summarized as follows.

Matching. We selected street, city, and zip codes as the blocking fields to construct query strings and evaluated various fuzzy string matching techniques for each field that contains randomly injected errors. Evaluated results suggest that the edit distance-based similarity matching method yields the best performance on matching both low-quality street base names and city names, and a q-gram-based method is the most robust method to erroneous zip code inputs. In terms of query fields, a matching query that hits against matching fields of street name, state, and zip code produced the best match rate while having a relatively short runtime among all possible query field permutations.

Ranking. We evaluated the ranking methods from four major categories: (1) attribute-based ranking, (2) term-frequency-based ranking, (3) classification-based ranking, and (4) hybrid ranking combines term-frequency and dense vector-based similarity. The attributed-based ranking methods with empirical weights of each address component from an existing geocoding system produced the best ranking performance. The random forest-based ranking method that ranks candidates based on the predicted two labels had a closed ranking performance to the attribute-based ranking method.

Parsing. We compared the parsing performance between two address parsers built upon a CRF model and a neural network architecture, respectively. Overall, the neural network-based address parser performed better than the statistical model-based address parsing for erroneous address input. For the CRF-based address parser, the selected domain-specific features only slightly improved the parsing performance, indicating the importance of feature engineering for a statistical

model-based NER parser. For the neural network-based address parsing, evaluation results suggest that word embeddings obtained from a context-based pre-trained language model with a weighted average strategy and the CRF-based tag decoder led to the best parsing performance. Compared to all other configurations, distributed input representations and optimizer have higher impacts on the parsing performance than the remaining evaluated configuration parameters, which are borrowed from the general NER task configuration and also led to the best address parsing performance.

3. A DEEP LEARNING APPROACH FOR ROOFTOP GEOCODING[1]

3.1 Introduction

Geocoding is the process of converting text-based description of a location into a pair of coordinates [92]. It has been widely used for spatial analysis in health, crime, and social research investigations [2, 3, 93]. However, relatively large spatial errors are still found in most modern geocoding systems due to the limitations of typical geocoding interpolation methods (i.e., Address Point, Address Polygon, and Street geocoding) and the availability and quality of the reference datasets they use [29, 28, 44, 42, 49]. Spatial errors in geocoding output are propagated through to the output data utilized in subsequent studies, thereby affecting the validity and accuracy of further data creation and the research conclusions of these studies [13]. For example, in air pollution exposure studies, two common geocoding error distances: 100 and 250 meters could lead to biased exposure classifications [14], and spatial errors produced by Street geocoding were proven to result in consistent overestimation for the number of exposed individuals [15]. In a cancer risk study, the spatial error for residential locations lower than 50 meters was required to conduct further analysis [16]. Among typical geocoding methods, Address Point geocoding which attempts to pinpoint the output to the centroid of a building rooftop has been shown to generate highly accurate results and is recommended for use if the data are available [41, 42]. However, due to the unavailability of rooftop reference datasets, this method is not widely used [49]. This leads to a question: Is there a way to automatically generate building rooftop geocoding outputs using highly available remotely sensed data? Recently, researchers have proposed different deep learning frameworks to extract objects from high resolution remote sensing imagery [94, 95, 96]. Several articles noted that aerial imagery can be used as supplementary material for producing geocoding reference dataset [97, 12, 49]. However, to date, a geocoding approach integrating object detection for generating rooftop centroid outputs has yet to be developed.

[1]Reprinted with permission from "A Deep Learning Approach for Rooftop Geocoding" by Zhengcong Yin, Andong Ma, Daniel W. Goldberg, 2019. *Transactions in GIS*, 23, 495– 514. ©2019 John Wiley and Sons

This work combines a typical geocoding workflow and building detection based on the Fast Region-based Convolutional Neural Network (Faster R-CNN) model to create a new geocoding approach. This allows geocoding systems to place outputs at the centroid of building rooftops without conducting typical geocoding interpolation methods or being completely limited by the availability of reference datasets. The contributions of this work are two-fold:

- From a methodology perspective: we develop a deep learning-based building rooftop extraction method and fully integrate it into the geocoding process. Then, we propose centroid candidate selection methods specific to this new geocoding workflow. Through an evaluation of 22,481 addresses, clear spatial error reductions resulting from the proposed approach in comparison to typical geocoding methods using the same reference dataset were observed, while maintaining a comparable match rate. Compared to Address Point/Polygon geocoding, our approach reduced the average spatial error by 4.96 meters (from 16.76 meters to 11.80 meters). Compared to Street geocoding, our approach reduced the average spatial error by 51.10 meters (from 130.46 meters to 79.36 meters). For different land-use types, our approach performed better on Low-density Residential and Commercial addresses than High-density Residential addresses.

- From an application perspective: the new geocoding approach provides the GIS community with new insights to leverage recent advances in the computer vision and deep learning domains for geospatial applications. For the geocoding community, our approach can be used to generate new address or Point-of-Interest (POI) reference datasets. For general GIS applications, the proposed approach can be extended to search different object locations with appropriate model setup and training.

The remainder of this chapter is organized as follows. Section 3.2 outlines the limitations of existing geocoding methods and recent advances in object detection. We detail the building rooftop centriod extraction method and how we integrate it into geocoding workflow in Section 3.3. In Section 3.4, we illustrate the evaluation metrics for our work and discuss the results. Finally, we

conclude this paper with potential avenues for future work in Section 3.5.

3.2 Related Work

3.2.1 Limitations of Current Geocoding Methods

Limitations of existing geocoding methods are well-studied in the literature. Although errors can be found in every component of a geocoding process, reference datasets and interpolation methods are still considered the main sources of spatial error [43, 20].

3.2.1.1 Street Geocoding

Street geocoding is the most widely used geocoding method, which utilizes street networks as reference sources. However, the outputs of Street geocoding are determined by the numeric value of the input and reference house numbers in a linear interpolation manner with three major assumptions: (1) the address existence assumption, (2) the uniform lot assumption, and (3) the parcel extension assumption [28]. According to USPS TIGER [2], house numbers are arbitrarily assigned to an address range, even though they do not exist in reality. All parcels on a street are assumed to have the same size and completely fill a street without corner offsets. A previous study attempted to tackle these issues by incorporating the number of buildings and the parcel size found online [30]. Arguably, this method relies heavily on the availability of auxiliary information. In addition to these assumptions, the offset distance from a street centerline to a building is pre-defined by geocoding systems without adjusting for variations across different places, which can also introduce errors into the output. [44]

3.2.1.2 Address Polygon Geocoding

Address Polygon geocoding uses geographic polygons representing one or more administrative units of geography as its reference dataset. To generate a geocoded output, a process for deriving a point from a polygon is required. Typically, there are three interpolation methods to generate such an output point: (1) the bounding box, (2) the mass of centroid, and (3) the weighted centroid

[2] https://www2.census.gov/geo/pdfs/maps-data/data/tiger/tgrshp2017/TGRSHP2017_TechDoc_Ch4.pdf

[20]. However, these three methods make the same assumption: the point derived from a polygon or a bounding box is the actual building location. This assumption was proven to work well for small-sized parcels but failed in large-sized parcels [29]. Additionally, for parcels with irregular shapes such as "L" or "S," this interpolation method cannot guarantee that the centroid is located inside a parcel [42]. Improving on this method, researchers used population density information to place the output at a more reasonable location [98]. However, this method typically only applies to large geographic areas, such as zip-codes, or counties where a population surface derived from a lower-level geography can be applied.

3.2.1.3 Address Point Geocoding

By definition, address points typically represent actual building locations [49]. The process of generating such a dataset, either by producing the centroid from a building footprint/parcel or field collection using the Global Positioning System (GPS) is costly in time and efforts [49, 42]. These costs help explain the limited coverage of these datasets. Because the reference data are already a point, no further interpolation is needed. The match rate and quality of the rooftop geocoding method completely rely on the quality of reference data [42]. Modern geocoding systems usually employ multiple sources of reference datasets [18]. This may lead one to question that which reference dataset should be used if different data sources are available.

3.2.2 Advances in Object Detection

Object detection is an image processing technology that detects semantic objects of a specific label in natural digital images. Girshick et al. [99] first proposed Region-based CNN (R-CNN) for object detection. Because there are numerous candidate regions, and each requires computation for feature extraction, the processing speed of R-CNN has been shown to be very slow. To address this, Girshick proposed a new training algorithm for R-CNN by utilizing a Region of Interest (ROI) pooling layer to train the whole network end-to-end, named as Fast R-CNN [100]. He et al. [101] investigated the utilization of a Region Proposal Network (RPN) to generate region proposals and innovated Faster R-CNN to have a much smaller time cost compared to R-CNN and Fast

R-CNN. Within the Faster R-CNN model, all learning schemes, including feature and proposal extraction, and bounding box regression, and classification are incorporated into a single deep learning network. These enhancements resulted in significant increases in processing accuracy and speed.

3.3 Workflow Formalization

The workflow of the proposed approach is depicted in Figure 3.1. Similar to typical geocoding, input addresses are parsed to generate a query string for the initial geographic context. We first conduct an *Exact Feature Matching* process using a deterministic matching method to retrieve geographic features that match the input house number. If there are zero matches, we revert to an upper geographic level to retrieve geographic features that match the input street name, a process known as *Fuzzy Feature Matching*. Next, the proposed approach constructs a *Searching Zone* using the retrieved geographic context for *Building Rooftop Centroid Extraction* to generate output candidates, the key difference from typical geocoding workflows.

3.3.1 Searching Zone

Since the spatial accuracy of the geographic context (i.e., address polygon or address point features) returned from *Exact Feature Matching* should be relatively high [49], we chose two *Benchmark Points* from them to construct the searching zone, as depicted in Algorithm 1. If the geographic context is a polygon feature, we selected two vertexes from this polygon that have the largest distance to be *Benchmark Points* so that we could calculate latitude and longitude differences per pixel as required by Algorithm 3. If a point feature has been retrieved, we use two endpoints of a 50-meter diagonal centered on this point feature as *Benchmark Points*. We consider this empirical radius (i.e., 25 meters) to be reasonable because a point feature is considered to represent a building centroid [49]. When both a point and a polygon feature are retrieved, we first attempt to derive *Benchmark Points* from the polygon feature since it can depict the actual parcel boundary for the input address.

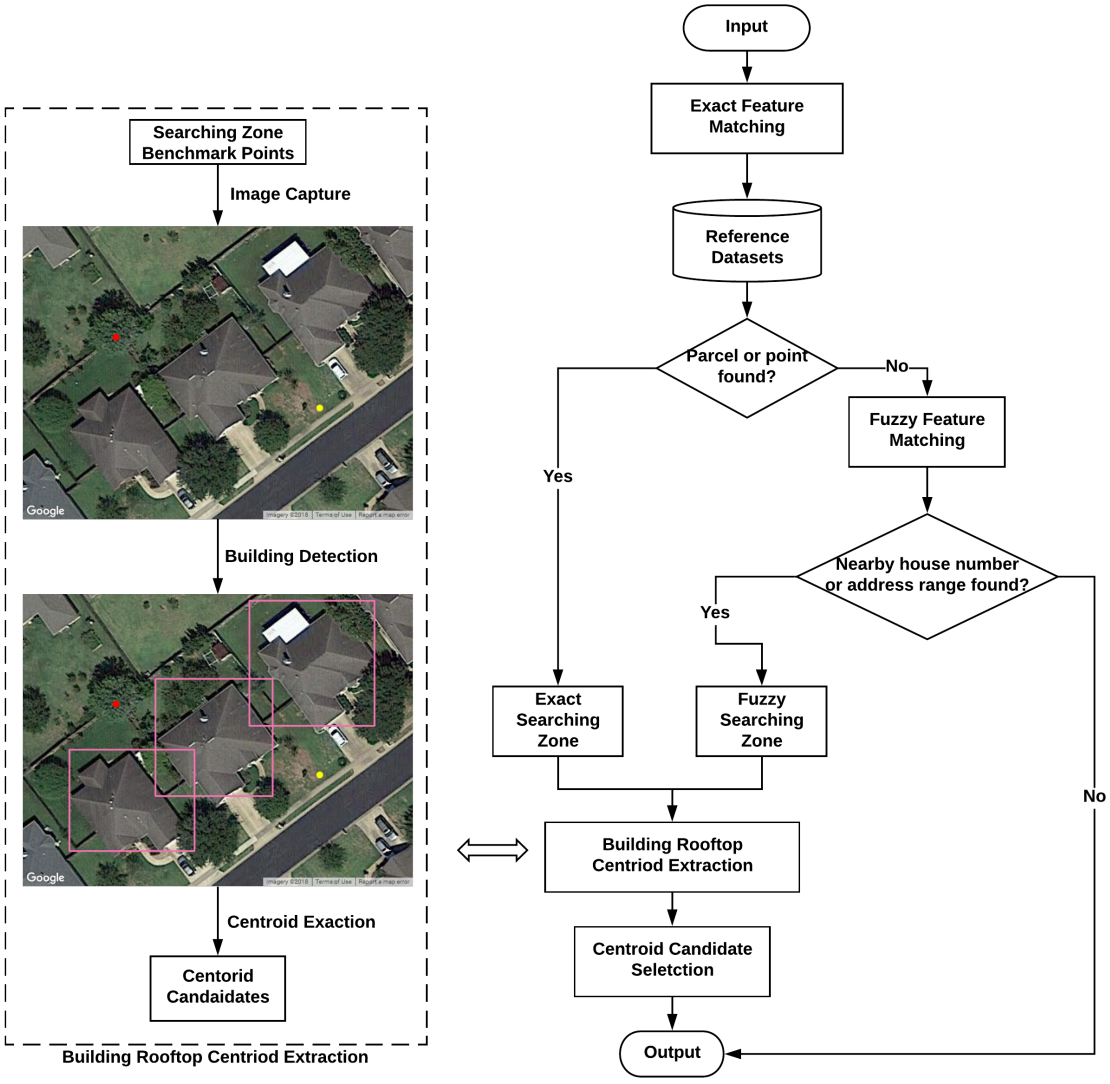


Figure 3.1: Workflow of the proposed approach

3.3.1.1 Fuzzy Searching Zone

If *Fuzzy Feature Matching* has been carried out, namely, an exactly matched point or polygon feature cannot be found, we build a *Fuzzy Searching Zone* that has the largest possibility of containing the input address. Inspired by using the "similar" house number to improve match rates [2], we select the geographic context with the minimum house number difference to the input address as *Benchmark Points* to bound this search zone, as depicted in Algorithm 2. We first attempt to identify the two house numbers that are closest to the input address and make sure they agree with

Algorithm 1: Identify two Benchmark Points of Exact Searching Zone

Input: collection of retrieved address point or address polygon features: $Dict_{addr}$;
Output: coordinates of two Benchmark Points: BP_1 and BP_2
$maxIndex$ = -1; $maxDist$ = -1;
if *polygon feature Polg found in* $Dict_{addr}$ **then**
  V_0 = randomly choose one vertex $V_i \in Polg$;
  for *each vertex* V_i *in* $Polg$ **do**
    if *Distance between* V_0 *and* V_i $>$ $maxDist$ ***and*** *latitude of* V_0 *!= latitude of* V_i ***and*** *longitude of* V_0 *!= longitude of* V_i **then**
      $maxDist$ = Distance between V_0 and V_i ;
      $maxIndex = i$;
    end
  end
  BP_1 = coordinates of V_0 ;
  BP_2 = coordinates of $V_{maxIndex}$;
  return BP_1 *and* BP_2;
end
if *point feature Pnt found in* $Dict_{addr}$ **then**
  BP_1 = coordinates of a point with length of 25 meters to Pnt and bearing of 45 degree;
  BP_2 = coordinates of a point with length of 25 meters to Pnt and bearing of 225 degree;
  return BP_1 *and* BP_2;
end

the parity (i.e., odd or even) of the input house number. According to USPS TIGER[3], the last two digits of an address range start from 00 and end with 99. Thus, we keep track of the valid address range for each identified house number to ensure it belongs to the block of the input address. If the nearest house number is outside the block of the input address, we use the nearest address range number that has the same parity of the input house number as the substitution. For example, given that the input house number is 1603, two nearby house numbers, 1515 and 1607, would be selected. We consider that house number 1515 would belong to another block as the valid address range for the input address should be 1600 to 1699. Since the range number 1601 has the smallest numeric difference to the input address, it is selected to bound the size of the searching zone (i.e.,

[3] https://www2.census.gov/geo/pdfs/maps-data/data/tiger/tgrshp2017/TGRSHP2017_TechDoc_Ch4.pdf

Algorithm 2: Identify two Benchmark Points of Fuzzy Searching Zone

Input: collection of retrieved address point, address polygon, and street line features: $Dict_{addr}$; house number of the input address: $hNum_{input}$
Output: coordinates of two Benchmark Points: BP_1 and BP_2
$hNumArr$ = []; $index_L$ = $index_R$ = -1;
for *each retrieved feature* **do**
    $hNumArr$.add (feature.house number);
end
$hNum\_Idx$ = Binary Search ($hNumArr$, $hNum_{input}$) ; // identify location of the input house number
while *hNumArr[hNum_Idx - 1].parity !=* $hNum_{input}$*.parity* **do**
    $index_L$ = $hNum\_Idx - 1$; // identify left-side nearby number which agrees parity
end
while *hNumArr[hNum_Idx + 1].parity !=* $hNum_{input}$*.parity* **do**
    $index_R$ = $hNum\_Idx + 1$; // identify right-side nearby number which agrees parity
end
BP_1 = Identify One-side Benchmark Point ($index_L$, $hNumArr$, $Dict_{addr}$);
BP_2 = Identify One-side Benchmark Point ($index_R$, $hNumArr$, $Dict_{addr}$);
return BP_1 *and* BP_2;
Function *Identify One-side Benchmark Point (idx,* $hNumArr$*,* $Dict_{addr}$*)*
    if *hNumArr[idx]/100 !=* $hNum_{input}$*/100 ;* // check if nearby house number is outside the block
    then
        return *closest endpoint of address range in* $Dict_{addr}$;
    end
    if *hNumArr[idx] is point* **then**
        return $Dict_{addr}$*[hNumArr[idx]].coordinates*;
    end
    if *hNumArr[idx] is polygon* **then**
        return $Dict_{addr}$*[hNumArr[idx]].centroid*;
    end
    if *hNumArr[idx] is street line* **then**
        return *closest endpoint of address range in* $Dict_{addr}$;
    end
    return *No Found*;

1601 - 1607). When a nearby house number is found in more than one reference dataset, we select the ideal reference data item based on the preferential ordering of reference data categories shown as optimal in a previous study: Point > Polygon > Street line [49]. The possible combinations of nearby features that can build a *Fuzzy Searching Zone* is summarized in Table 3.1.

Table 3.1: Possible nearby features for Benchmark Points of Fuzzy Searching Zone

Nearby Features	Possible Scenarios
Two house numbers	Two nearby house numbers are found in polygon or point reference datasets.
One house number, one address endpoint	(1) Only one nearby house number is found in point or polygon reference datasets and the other nearby feature is retrieved from street line reference datasets; (2) One of two retrieved nearby house numbers belongs to the block of the input address.
Two address endpoints	(1) Only address range endpoints are found; (2) Two retrieved nearby house numbers both do not belong to the block of the input address.
N/A	No nearby house numbers are found in polygon or point reference datasets and the matched address range is found in street line reference datasets.

3.3.2 Building Detection and Centroid Extraction

3.3.2.1 Model Setup and Data Training

We collected 250 aerial RGB images with the size of 640 $\times$ 480 pixels in two cities in Texas from Google Earth[4], and labeled all rooftops on each image. We also expanded the training samples by flipping images both horizontally and vertically and conducting 90-degree image rotation.

We adopted the Faster R-CNN model for building rooftop detection owing to its performance and reliability [101]. ResNet-50 [102] was used as the base deep neural network, which was pre-trained by ImageNet[5]. Our Region Proposal Network (RPN) consisted of three convolutional layers. The input layer was assembled by 512 kernels with size of 3 $\times$ 3. The class prediction layer was created by 9 kernels with size of 1 $\times$ 1. The regression layer contained 9 $\times$ 4 kernels with size of 1 $\times$ 1. The total number of epochs and the length of each epoch were set as 50 and 400, respectively. With this setup, model training was completed in 12.5 hours.

[4] `https://www.google.com/earth/`
[5] `http://image-net.org/index`

3.3.2.2 Building Detection and Centroid Extraction

As shown in Figure 3.1, with the help of two *Benchmark Points* (i.e red/yellow dot), we captured satellite images covering the building in question using Google Maps [6]. Then, the Faster R-CNN model output a rectangular bounding box for each detected building rooftop on the input images. Since the bounding box is presented as the pixel locations (i.e., row and column) of top-left and bottom-right corners: P_{tl} and P_{br}, it becomes possible to calculate each bounding box's centroid: P_c to represent a building location. We take advantage of two *Benchmark Points*: BP_1 and BP_2 by detecting their row/column by their own pre-assigned RGB values. Because lat/lng of these two *Benchmark Points* are known, we can obtain the coordinate difference per pixel for the input image. Finally, we can calculate the lat/lng of P_c using its pixel location related to the input image's top-left corner, which is (0, 0).

Algorithm 3: Building Rooftop Centroid Extraction

Input: geographic coordinates of two $BenchmarkPoints$: BP_1, BP_2 and their RGB values;
Output: geographic coordinates of the extracted centroid point: P_c
Detect BP_1, BP_2 by RGB values, return pixel location (row, column);
Detect building rooftop bounding box(es), return pixel location of top-left P_{tl} $(row, column)$ and bottom-right P_{br} $(row, column)$;
lat_per_pixel = $|BP_1.lat - BP_2.lat|$ / $|BP_1.row - BP_2.row|$; `// Calculate lat/lng per pixel`
lng_per_pixel = $|BP_1.lng - BP_2.lng|$ / $|BP_1.column - BP_2.column|$;
$I_{tl}.lat$ = $P_1.lat + lat\_per\_pixel \times P_1.row$; `// Calculate lat/lng of image's top-left corner`
$I_{tl}.lng$ = -(-$P_1.lng + lng\_per\_pixel \times P_1.column$);
for *each bounding box(es)* **do**
    $P_c.row$ = ($P_{tl}.row$ + $P_{br}.row$)/2 ; `// Calculate row/column of` P_c
    $P_c.column$ = ($P_{tl}.column$ + $P_{br}.column$)/2;
    $P_c.lat$ = $I_{tl}.lat - lat\_per\_pixel \times P_c.row$; `// Calculate lat/lng of` P_c
    $P_c.lng$ = - (-$I_{tl}.lng - lng\_per\_pixel \times P_c.column$);
    return P_c
end

[6] `https://developers.google.com/maps`

3.3.3 Centroid Candidate Selection

3.3.3.1 Candidate Validation

To facilitate candidate selection, we use a multi-step procedure based on topological relationships to filter out invalid candidates.

Step 1: The first step is to make sure candidates are located in reasonable areas. Since each searching zone is the area that has the largest possibility of containing the input address, we conduct a point-in-polygon analysis to filter out candidates that are outside the constructed searching zone.

Step 2: The second step is to check if the candidate agrees with the address parity rule (i.e., odd and even house numbers are located on the separate sides of a street). To do so, we propose a *Clockwise Checking* method to select candidates that obey this rule.

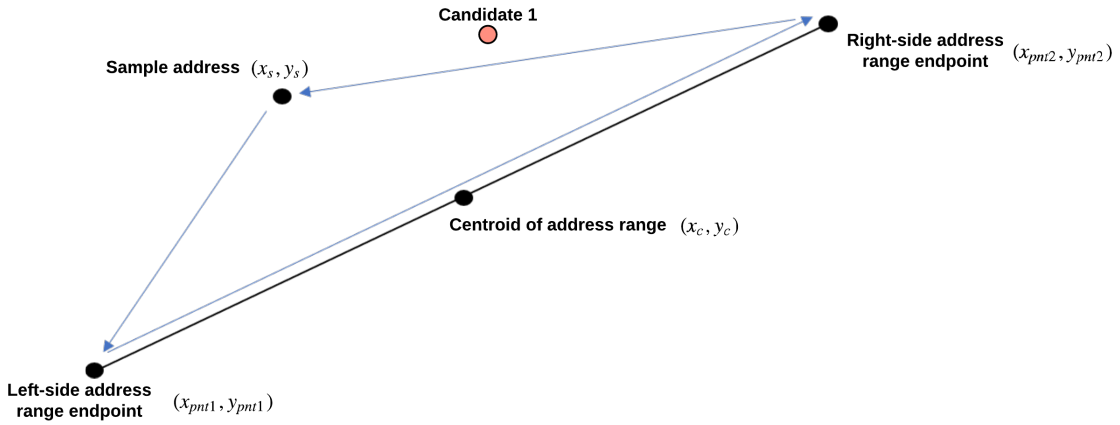


Figure 3.2: Clockwise checking method to check address parity

$$Endpoint = \begin{cases} Left\ side & \text{If } x_{pnt1} < x_c \\ Right\ side & \text{If } x_{pnt2} > x_c \end{cases} \tag{3.1}$$

$$\begin{vmatrix} x_s & y_s & 1 \\ x_{pnt1} & y_{pnt1} & 1 \\ x_{pnt2} & y_{pnt2} & 1 \end{vmatrix} = \begin{cases} > 0 & \text{If Counter Clockwise} \\ < 0 & \text{If Clockwise} \end{cases} \tag{3.2}$$

As shown in Figure 3.2, with the help of Equation 3.1, we can determine the left- and right-side endpoints of an address, denoted as (x_{pnt1}, y_{pnt1}) and (x_{pnt2}, y_{pnt2}). Next, we randomly sample a nearby address from the reference dataset, named as *Sample Address*, and connect these points as follows: *Sample Address* -> Left-side endpoint -> Right-side endpoint. Finally, we determine if this path is clockwise or counter-clockwise using Equation 3.2. To agree with the address parity rule, all valid candidates should have the same path, if the input address and *Sample Address* have the same house number parity.

Step 3: The last step is to test if a candidate is a false positive match, which means that it is actually another address that exists in the reference datasets. We first conduct a K-Nearest Neighbor (K-NN) query to retrieve 10 nearby addresses from the reference dataset. Then, we compare the distances between a candidate and the 10 nearby addresses to a threshold distance of 3 meters. In the case when the smallest distance is less than 3 meters and their address descriptions are different, the candidate is determined as a false positive match. We consider this threshold distance to be reasonable, as it is much smaller than the conventionally employed off-set distance of 15 meters [11].

3.3.3.2 Candidate Selection

The candidate selection process determines the best candidate for the final output. Because the two types of searching zones result in different levels of confidence, the candidate selection process works differently for each.

Exact Searching Zone: If multiple candidates exist in an *Exact Searching Zone*, it can be a multi-building residential structure (e.g. a house with a detached garage), an apartment complex, or a commercial complex. However, land-use information is difficult to obtain from reference datasets. To deal with possible scenarios, the selection procedure is executed as follows:

- If the searching zone is constructed by address polygon reference data, we utilize the threshold value: 5 as the minimum number of candidates to differentiate an apartment complex from a residential parcel. This value was determined by a manual review of local apartment complexes. If the number of candidates is less than 5, the input address is determined to be a residential parcel, and the candidate with the largest bounding box is selected because we assume that the main residential building would typically be larger than other structures (such as a detached garage). Otherwise, we consider the input address to be an apartment complex or a commercial complex. Typically, available reference datasets do not have unit number information for each building, which is a consistent difficulty for geocoding systems [15]. Thus, we use the centroid of a convex hull assembled from all detected rooftops as the output. This centroid is weighted by the location of the buildings in a parcel, which is considered to derive a more reasonable output than a geometric centroid [92, 98].

- If the searching zone is constructed by address point reference data, we empirically select the candidate which is closest to this reference data as the best candidate. Owing to the definition of address point reference data [49], the nearest candidate to this point feature is the most likely to be the best candidate.

Fuzzy Searching Zone: We propose a *House Number Difference Ranking* method to select the best candidates that exist in a *Fuzzy Searching Zone*, as illustrated in Algorithm 4. Intuitively, the distance from a candidate to a benchmark address reflects the numeric difference between these house numbers. Therefore, an important step of the algorithm is to determine a benchmark address.

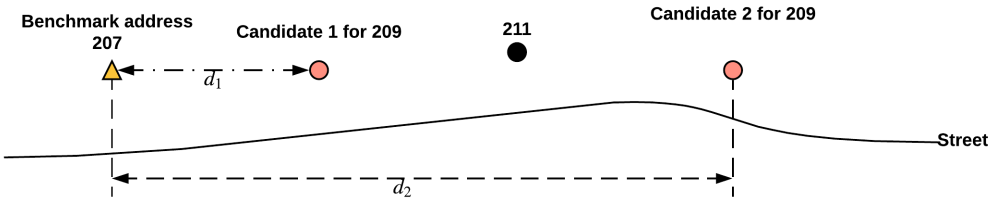


Figure 3.3: House number difference ranking to select the best candidate

Because two *Benchmark Points* that bound a *Fuzzy Searching Zone* have the house numbers and geographic locations which are near to the input address, we select one of them as the benchmark address. First, the *Benchmark Point* that has a smaller house number difference than the input house number is used as the benchmark address. If both *Benchmark Points* have the same house number difference to the input address, we choose the one that comes from a high-quality reference source as follows: Point > Polygon > Street line. The other key step is to determine the best candidate from the candidate list sorted by distances from each candidate to the benchmark address in an ascending order. As illustrated in Figure 3.3, considering the house number parity, the input address 209 is only one magnitude away from the benchmark address 207, thereby making it the candidate with the smaller distance d_1 to the benchmark address more likely to be the candidate for the input address. It is possible that the benchmark address 207 has been included in the candidate list. To check this situation, we compare the distance between the benchmark address and the first candidate in the sorted list to the same value used for testing a false positive match. If this distance is less than 3 meters, we would choose the second candidate in the candidate list as the final output. Otherwise, we would choose the first one.

3.4 Experiments and Results

3.4.1 Experimental Setup

Our evaluation primarily seeks to answer three specific questions: (1) What is the match rate of our approach; (2) To what degree our approach can reduce spatial error; and (3) Is the performance of our approach sensitive to land-use types.

The reference data used in the experiment included all three address models: address point, address polygon, and street line. Address point reference data were extracted from Navteq[7]. Address polygon reference data were obtained from the Boundary Solutions parcel database. Street line reference data were generated from the US Census Bureau TIGER/Lines[8]. These reference data sources were chosen because they represent the typical types and qualities of reference data

[7] `https://www.here.com/en/navteq`

[8] `https://www.census.gov/geo/maps-data/data/tiger-line.html`

Algorithm 4: House Number Difference Ranking

Input: collection of candidates: $Dict_{candidate}$; the input house number: $hNum_{input}$; two *Benchmark Points*: BP_1 and BP_2;
Output: coordinates of the best candidate $Coord_{best}$
$hNum_{benchmark}$ = 0; $distArr$ = []; $Ret_{candidate}$ = {};
if $hNum_{input} - BP_1.house\ number! = hNum_{input} - BP_2.house\ number$ **then**
    $hNum_{benchmark}$ =
     $\min(hNum_{input} - BP_1.house\ number, hNum_{input} - BP_2.house\ number)$;
    $Addr_{benchmark}$ = ***Benchmark Point*** with $hNum_{benchmark}$;
end
else
    $Addr_{benchmark}$ = ***Benchmark Point*** selected by the order: Point > Polygon > Street line;
end
$Coord_{benchmark}$ = coordinates of $Addr_{benchmark}$;
for *each candidate* i **do**
    $Dist_i$ = Distance between $Coord_{benchmark}$ and $Coord_i$;
    $distArr$ **add** $Dist_i$;
    $Ret_{candidate}[Dist_i] = Coord_i$;
end
Sort $distArr$ by distance in ascending order;
if *the first candidate is the benchmark address* **then**
    $hNum_{diff}$ = $hNum_{input}$ - $hNum_{benchmark}$; `// To avoid the first candidate being selected`
end
else
    $hNum_{diff}$ = $hNum_{input}$ - $hNum_{benchmark}$ - 1;
end
$Coord_{best}$ = coordinates of $Ret_{candidate}[distArr[hNum_{diff}]]$;
return $Coord_{best}$;

sources routinely used by researchers and organizations [28]. The test dataset was obtained from the city of College Station and contains 22,481 building centroid coordinates associated with address descriptions across three land-use types: Low-density Residential, High-density Residential, and Commercial.

To conduct the evaluation, the proposed approach was implemented as a web service using Node.js[9] and all reference data were indexed in the Elasticsearch[10] NoSQL database framework.

[9] `https://nodejs.org`
[10] `https://www.elastic.co/products/elasticsearch`

We geocoded the test dataset using our approach (denoted as *DLRooftop*), the Texas A&M Geocoding Services[11] (denoted as *Base Reference*) and Google's Geocoding API[12] (denoted as *Google*), respectively. Then, outputs produced by each system were compared in terms of match rate and spatial accuracy (i.e., the distance between geocoded data and the true value), which is consistent with other studies [18, 103, 104]. It is worth noting that the proposed approach and the Texas A&M Geocoding Services utilized the same reference dataset, whereas Google maintains its own reference datasets of unknown number, type, or quality, though these are typically assumed to be numerous and of very high quality. Given the non-normal distribution of spatial error and the relatively large sample size (n >30), a Wilcoxon signed rank test and a Student's *t*-test on the original and the log-transformed spatial error was performed to determine the significance of spatial errors are reduction, respectively [18, 105].

3.4.2 Results and Discussions

3.4.2.1 Match Rate

Table 3.2: Match Rates

System	Match Level	Match Rate (%)
DLRooftop	Rooftop	82.18
DLRooftop	Unmatched	17.82
Base Reference	Building	80.83
Base Reference	Street	2.86
Base Reference	Unmatched	16.31
Google	Rooftop	95.05
Google	Other	4.87
Google	Unmatched	0.8

Table 3.2 lists match rates from different geocoding systems. *Google* achieved the highest match rate. In comparison to *Base Reference*, the proposed approach slightly reduced the overall

[11] `http://geoservices.tamu.edu/`
[12] `https://developers.google.com/maps/documentation/geocoding/start`

match rate by 1.51%. The main reason for this is that the proposed approach increases the granularity of the match level. Each output produced by the approach must match a building rooftop, a higher spatial level, with matches of a lower accuracy being thrown away. In total, our approach produced 82.18% rooftop geocoded results, improving the building level match rate by 1.35% from 80.83%. In contrast, 2.86% of geocodes were in the *Base Reference* only matched the street level, a lower spatial accuracy level. Such trade-offs between the match rate and spatial accuracy have been shown in previous studies as well [49]. Overall, our approach maintained a comparable match rate to that of typical geocoding methods, even though it produced outputs with higher geographic levels of matches (rooftop vs. street level).

3.4.2.2 Overall Spatial Accuracy

When compared to the match rate, spatial accuracy is considered a more important metric to demonstrate the quality of a geocoding method [44, 11, 49, 18, 70]. To assess spatial accuracy, we filtered out unmatched records from each system and grouped outputs by feature matching scenarios, which refer to different typical geocoding methods. The descriptive statistics (Table 3.3) and the cumulative distribution of spatial error (Figure 3.4) show that the proposed approach significantly reduced spatial error compared to typical geocoding methods (i.e., Base Reference) in both feature matching scenarios.

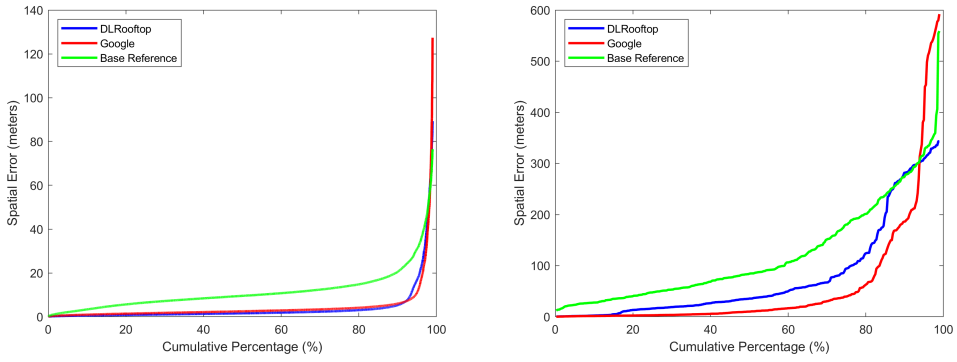


Figure 3.4: Cumulative distribution of spatial error in Exact Feature Matching (Left) and Fuzzy Feature Matching

(Right) scenarios

Table 3.3: Descriptive statistics for overall spatial error (meters) in Exact and Fuzzy Feature Matching scenarios

	Exact Feature Matching			Fuzzy Feature Matching		
Statistics	DLRooftop	Base Reference	Google	DLRooftop	Base Reference	Google
25th	0.96	6.49	1.60	16.31	47.98	2.94
50th	1.58	9.54	2.49	35.61	84.46	10.11
75th	2.60	13.45	3.69	94.51	180.56	39.73
Average	11.80	16.76	951	79.36	130.46	62.46
p (t-test)		< 0.001	< 0.001		< 0.001	< 0.001
p (wil. sign)		< 0.001	< 0.001		< 0.001	< 0.001

In Table 3.3, compared to *Base Reference* in the *Exact Feature Matching* scenario (i.e., Address Point/Polygon geocoding method), our approach reduced the average and the median spatial error by 4.96 meters (from 16.76 meters to 11.80 meters) and by 7.96 meters (from 9.54 meters to 1.58 meters), respectively. Our approach also outperformed *Google* in this scenario, producing outputs with smaller spatial error, even though we did not have access to their luxury reference dataset. The spatial error reduction produced by our approach was shown to be statistically significant in comparison to these two systems ($p < 0.001$). Such improvement was gained because our approach can place the output closer to the true building location with the help of rooftop detection, as shown in Figure 3.5. The left-side of Figure 3.5 shows that our approach and *Google* can place the output at the centroid of rooftop which is close to the true location, whereas the output of *Base Reference* was far away from the actual building location, depending on the spatial accuracy of reference data. On the right-side of Figure 3.5, *Base Reference* output the centroid of the parcel, since address point reference data could not be found. However, the parcel centroid may not be the building's location, especially for a large-sized parcel, which is known as a major drawback of Address Polygon interpolation methods [29]. Because our approach can detect building rooftops, we do not need to conduct this interpolation to generate an output.

As can be seen in the *Fuzzy Feature Matching* scenario of Table 3.3, *Google* produced the most accurate results because it resulted in a rooftop match level for input addresses. These addresses could not be found in the address point or address polygon reference datasets employed by our

Figure 3.5: Output for small-size (Left) and large-size parcels (Right) in Exact Matching scenarios

Figure 3.6: Output for a Low-density Residential address (Left) and a Commercial address (Right) in Fuzzy Feature Matching scenarios

approach and *Base Reference*, therefore *Base Reference* performed the Street geocoding method. Unlike Street geocoding which conducts its linear interpolation only on address ranges, our approach is able to estimate the input address location utilizing nearby buildings (Algorithm 4). As shown in the left-side of Figure 3.6, for an input address 2820, two nearby house numbers (i.e., yellow triangles) with the same parity as the input address were retrieved, and 2818 was determined as the benchmark address due to the smaller numeric difference related to the input address. Since 2820 is only one magnitude away from the benchmark address, we chose the nearest candidate as the final output. The right-side of Figure 3.6 demonstrates that this method also applied

to Commercial addresses, even though only one nearby house number has been identified. In contrast, outputs of *Base Reference* (i.e., orange dots) were far away from the true data for these two cases due to the limitations of Street geocoding. To this end, our approach reduced the average and median spatial error of *Base Reference* by 51.10 meters (from 130.46 meters to 79.36 meters) and by 48.85 meters (from 84.46 meters to 35.61 meters), respectively. Such spatial error reduction resulting from our approach was seen to be statistically significant ($p < 0.001$).

These examples in Figures 3.5 and 3.6 also demonstrate that our approach has the potential to generate new address point level reference data, when the original reference data are not precise enough or are not available. However, it is worth noting that the quality of reference data could impact the output quality, as it may suggest an incorrect searching zone. For example, the initial geographic context for the input address: "901 Texas Ave" resolved at the location of "901 S Texas Ave", because this reference data missed the directional component of the input address. Thus, the final output was also pinpointed at the incorrect location. This can be one reason for the large spatial error variance observed in Figure 3.4 and Table 3.3, namely, where the curve for each geocoding system result overlaps near the 100% mark and the large difference between the mean and the median spatial error produced by our approach. The main reason for this phenomenon is due to the geocoding performance results varying by different land-use types, which is observed in previous studies as well [49, 97, 29].

3.4.2.3 Spatial Accuracy for Different Land-use Types

To further explore the performance variations seen in the results of this testing, we grouped geocoded results by land-use types. The descriptive statistics of spatial error for each land-use type in the *Exact* and the *Fuzzy Feature Matching* scenarios were summarized in Tables 3.4 and 3.5, respectively. Both tables show a similar pattern for each feature matching scenario: our approach performed better for Low-density Residential and Commercial addresses than for High-density addresses in terms of reducing spatial error from *Base Reference*. As can be seen from these two tables, for Low-density Residential addresses, our approach reduced the mean spatial error of *Base Reference* by 4.95 meters (from 13.42 meters to 8.47 meters) in the *Exact Feature*

Matching scenario and by 45.57 meters (from 81.71 meters to 36.14 meters) in the *Fuzzy Feature Matching* scenario, respectively. For Commercial addresses, our approach reduced the mean spatial error of *Base Reference* by 9.40 meters (from 85.59 meters to 76.19 meters) in the *Exact Feature Matching* and by 152.28 meters (from 215.64 meters to 63.36 meters) in the *Fuzzy Feature Matching* scenario, respectively. For High-density Residential addresses, our approach delivered output with the similar mean spatial error to *Base Reference* in each feature matching scenario. The statistical analysis also showed that in comparison to *Base Reference*, the spatial error reduction resulting from our approach for Low-density Residential addresses and Commercial addresses were significant but not for High-density Residential addresses in both feature matching scenarios.

Table 3.4: Descriptive statistics of spatial error (meters) across land-use types in the *Exact Feature Matching* scenario

	Low-density Residential (n=17,138)			High-density Residential (n=333)			Commercial (n=487)		
Statistics	DLRooftop	Base Reference	Google	DLRooftop	Base Reference	Google	DLRooftop	Base Reference	Google
25th	0.94	6.49	1.57	3.35	4.32	2.89	2.18	11.42	2.96
50th	1.53	9.47	2.45	7.94	7.76	4.95	5.64	22.04	6.82
75th	2.45	13.14	3.55	21.85	20.14	7.88	21.25	41.37	19.62
Average	8.47	13.42	984	89.41	88.04	10.72	76.19	85.59	409.85
p (t-test)		< 0.001	< 0.001		0.43	< 0.001		< 0.001	< 0.001
p (wil. sign)		< 0.001	< 0.001		0.023	< 0.001		< 0.001	0.020

Such variations in the performance can be explained by the building distribution and the details of reference datasets for different land-use types. The reference dataset of Low-density Residential addresses usually contains house number information for each parcel or each building. Such information helps our approach to precisely pinpoint the correct parcel and detect the rooftop centroid without being affected by the parcel size or nearby buildings in the scenario of *Exact Feature Matching*, as shown in Figure 3.5. The regular distribution of Low-density Residential buildings, shown in Figure 3.6, helps to estimate the input address location using nearby house numbers in the *Fuzzy Feature Matching* scenario. Similar to Low-density Residential buildings, Commer-

Table 3.5: Descriptive statistics of spatial error (meters) across land-use types in the *Fuzzy Feature Matching* scenario

	Low-density Residential (n=240)			High-density Residential (n=77)			Commercial (n=59)		
Statistics	DLRooftop	Base Reference	Google	DLRooftop	Base Reference	Google	DLRooftop	Base Reference	Google
25th	12.11	36.86	2.37	109.10	158.52	40.14	21.28	69.86	3.37
50th	22.63	63.43	6.06	272.83	221.55	181.98	56.36	109.39	6.15
75th	40.88	99.45	19.82	305.09	293.13	380.29	89.37	211.01	13.92
Average	36.14	81.71	16.94	226.33	217.13	228.61	63.36	215.64	30.79
p (t-test)		< 0.001	< 0.001		0.274	0.006		< 0.001	< 0.001
p (wil. sign)		< 0.001	< 0.001		0.395	0.027		< 0.001	< 0.001

cial buildings are aligned regularly in a large-sized parcel, and reference datasets for Commercial addresses, known as POI datasets are often well-detailed. Thus, the spatial error reduction of Commercial addresses by our approach was comparable to Low-density addresses. In contrast, reference datasets for High-density Residential addresses (i.e., apartment complexes or multi-unit buildings) are less detailed (i.e., they do not contain house number or unit number information for each building) [15], and the building distribution can be ambiguous. Even though we can detect the

Figure 3.7: Two output examples for High-density Residential addresses

number of buildings in a complex, it is difficult to assign an address to each building in the parcel using a generic building distribution rule. Thus, our approach outputs the centroid (i.e., blue dot) of a convex hull (i.e., red outline) composed from each of the detected rooftops, as shown in the

left-side of Figure 3.7. Even though this centroid is weighted by building location, it still shares the same drawback of typical Address Polygon interpolated methods. For multi-unit apartments, a single building contains multiple addresses. As shown in the right-side of Figure 3.7, each address only occupied a small portion of a rooftop. However, the *Building Centroid Detection* process output only one centroid per rooftop, generalizing multiple address locations as one. These factors explain why the spatial error in the output from our approach was similar to *Base Reference* in both scenarios for this land-use type.

3.4.2.4 Sources of Uncertainty and Limitations

To increase the transparency of our proposed approach, provide insight for studies that utilize our results, and motivate future improvement, we next summarize the uncertainties and limitations that exist in the workflow.

First, *Building Rooftop Extraction* played a critical role in determining the final output. This process relied on aerial images obtained from online map services. Thus, the uncertainty of the map source from the perspectives of the image quality and temporal accuracy would impact the quality of geocoded outputs, resulting in a lower match rate and a larger spatial error. The detection process could be limited by the training dataset. Arguably, our training dataset was relatively small and we only labelled building rooftops, omitting other common objects appearing in human-housing areas such as swimming pools or vehicles. Even though the performance of the Faster R-CNN has been demonstrated in the past [101], it could not guarantee that all buildings in the input images would be precisely detected. In addition, the output bounding box from the detection model resulted in a planar surface being used to calculate the centroid coordinates, which may limit the coordinate accuracy.

Second, in addition to the reference data quality, the address parsing and the feature matching processes could also lead to incorrect initial geographic contexts for building rooftop detection, eventually impacting the final output. In this work, we only employed the deterministic feature matching approach for the geographic context retrieval, and input addresses were well-formatted. To deal with more sophisticated input addresses and to improve the accuracy of the initial geo-

graphic context retrieval, more effort should be devoted to the development of address parsing and feature matching algorithms.

Third, while the candidates selection strategy is specific to land-use types, it still has limitations. Since the reference datasets used here did not provide land-use information, we differentiated land-use types by the number of buildings chosen from empirical testing. This may limit the accuracy of our judgement. For apartment complex addresses, without a rule to assign a house number to each building, we can only weight the output by the nearby building location, which largely limits the output accuracy. When multiple candidates align on a street, we determined the best candidate by candidate locations related to a benchmark address and the house number difference. Even though this algorithm does not use assumptions that are made by Street geocoding, it also relies on the distributions of buildings and house numbers. If the difference between the input house number and the benchmark house number is too large, or the building distribution is irregular, this method may lead to a low spatial accuracy output.

3.5 Conclusions

The output quality of geocoding systems has been limited by typical geocoding interpolation methods and reference datasets for long time. In this chapter, we present a novel geocoding approach utilizing object detection based on deep learning framework to generate rooftop geocoding output. Specifically, we introduce a method to obtain coordinates of building rooftop centroids using the Faster R-CNN model from aerial imagery and integrate this method into a geocoding process. Then, we develop a series of centroid candidate selection methods specific to this workflow. Compared to typical geocoding methods, clear spatial error reduction resulting from our approach is observed, while maintaining a comparable match rate. The performance variation of our approach for different land-use types is found due to the building distributions and detail of reference datasets.

4. A PROBABILISTIC APPROACH FOR IMPROVING REVERSE GEOCODING OUTPUT[1]

4.1 Introduction

Reverse geocoding comprises the process of converting a pair of coordinates to the information contained in that location [106, 8]. Recent decades have seen a significant evolution of such transition from machine-readable GPS coordinates to human-readable location information. On the one hand, recent advances in ubiquitous GPS-enabled devices have sparked the need to link spatial coordinates to place descriptions, making reverse geocoding prevalent in navigation and routing services [107]. On the other hand, as more information such as Point-of-Interests (POIs) and human activities is added to locations, reverse geocoding has been used for retrieving information to support crime analysis, site selection, and human trajectory retrieval [108, 109, 110, 111]. These applications not only expand the content of reverse geocoding output but also create a huge demand for better reverse geocoding output quality to benefit end-users or subsequent studies. For example, a precise location description obtained from reverse geocoding could help first responders understand where you are in emergencies.

Modern commercial map providers such as Google Maps[2], HERE WeGo[3], Bing Maps[4], and Esri[5] offer reverse geocoding services to the public. However, the outputs from these reverse geocoding systems suffer from three challenges. First, current solutions typically rely on distance only for candidate selections, failing to consider spatial topology relationships. While it is possible that the returned candidate is the closest to the input points, it may be located on the opposite side of the street. Such a candidate can lead to problems with passenger pickups (for example) as a driver may need to make a U-turn to arrive at designated locations.

[1] Reprinted with permission from "A Probabilistic Approach for Improving Reverse Geocoding Output" by Zhengcong Yin, Daniel W. Goldberg, Tracy A. Hammond, Chong Zhang, Andong Ma, Xiao Li, 2020. *Transactions in GIS*, 24, 656– 680. ©2020 John Wiley and Sons

[2] `https://www.google.com/maps`

[3] `https://wego.here.com`

[4] `https://www.bing.com/maps`

[5] `https://developers.arcgis.com/rest/geocode/api-reference/geocoding-reverse-geocode.htm`

Second, current output metadata do not describe how candidates are ranked. Candidates in output list are ranked in an uninformative manner, as the current metadata descriptions only include distance to the input coordinates and the geographic match level, making the ranking mechanism unclear. Therefore, subsequent processes have limited information that could be used to choose the best candidate from output lists. Modern reverse geocoding systems have started to use multiple reference data sources and additional information to suggest the best candidates. Therefore, a consistent ranking mechanism is needed to order candidates drawn from different reference sources to improve location suggestions.

Third, current approaches have not utilized input GPS uncertainty as a contributing factor for determining output or for reporting error/uncertainty. Prior work has suggested uncertainty introduced by GPS sensors is unavoidable [112, 113]. However, such uncertainty is omitted from current reverse geocoding solutions, meaning that a correct output location might be easily excluded from a candidate set of potential outcomes. Given more LBS applications make use of mobile-phone based GPS with varying degrees of accuracy, considering GPS uncertainty become a necessity.

Given these observations, we argue that reverse geocoding output should (1) be spatially close to and topologically correct with respect to the input location (2) contain multiple suggestions ordered by the likelihood of being the best candidate, and (3) incorporate input GPS uncertainties. To meet these criteria, we proposed a probabilistic framework to improve the quality of reverse geocoding output. The contributions of this work are as follows:

- We present a reverse geocoding workflow that leverages spatial topological relationships between existing address models (i.e., address point, address parcel, street segment) for candidate selection, moving beyond simple distance-only measures to differentiate candidates. This results in comparable best candidate quality (i.e., one based on correct first candidate and number of errors) in comparison to that offered by four commercial reverse geocoding systems: Google, Microsoft, Esri, Here, and one open-access reverse geocoding system: Texas A&M GeoServices.

- We introduce a ranking mechanism to quantify topological relationships and distances to address candidates with a uniform quantification. Instead of only returning the single best candidate, this approach enables an address candidate sorting, which provides end-users with a clearly ranked set of potential candidates. Evaluation results indicate that candidates ordered by the proposed method mimic human spatial cognition more effectively in comparison to existing ranking approaches.

- We propose an algorithm for incorporating input GPS uncertainties into the output results. We derive the statistical surface for input GPS coordinates, combine it with the candidates from the proposed ranking mechanism, and utilize this uncertainty as part of the computational process. Experimental results show that candidate lists generated by the proposed approach successfully capture and reflect input uncertainties.

The remainder of this chapter is organized as follows. Related work is described in Section 4.2. In Section 4.3, we describe the components of the proposed reverse geocoding workflow. Section 4.4 and Section 4.5 introduces a weight-based quantification method for ordering each candidate and the algorithm that propagates the input GPS uncertainty to output candidate lists, respectively. In Section 4.6, we illustrate the evaluation metrics for our work and discuss the results. We conclude this paper with potential avenues for future work in Section 4.7.

4.2 Related Work

Efforts to improve reverse geocoding have been carried out in both industry and academia. Industrial work has primarily focused on system performance and system optimization such as caching addresses and POIs within bounding areas to improve query performance [34]; tuning contextual output based on approximations of reference points [114]; and integrating street segments data with street points data [36]. A performance related study in academia explored the feasibility of spatial partition methods such as the Global Subdivision Model (GSM) and Geohash to reduce the query latency of reverse geocoding systems [35].

In terms of improving output quality, the majority of the existing solutions achieve this by

mining a variety of data sources, including Geo-tagged text, photos, and users' demographic data [37, 38, 39, 40]. These works focused on weighting POIs by modeling geographic footprints with demographic data, human activity, and temporal signatures, respectively. Other works have attempted to distort distance measures to nearby POIs as a function of time [106], or directly embed POIs based on distributional semantics [115]. Arguably, the significance of these works could be limited by the availability of additional supplementary information and the quantification of each candidate before considering these supplementary information.

While reverse geocoding is widely used across domains, relatively few works have surveyed the output quality produced by existing reverse geocoding systems [116]. To our knowledge, to date, no work has explicitly studied the metadata descriptions of reverse geocoding. Since few implementation details of commercial systems have been revealed, these systems are treated as a "black box" when they are employed in different application scenarios. Current efforts regarding quality and metadata reporting exist primarily in the realm of geocoding. The quality of geocoding systems is typically quantified by match rate and spatial accuracy [49, 26, 70]. Geocoding interpolation techniques (e.g., street geocoding, address polygon geocoding, address point geocoding, and object detection approach) have been evaluated for different land-use locations (i.e., urban and rural areas) or different address types (i.e., postal address and P.O. Box) [29, 44, 45, 46]. [19] proposed a framework for evaluating geocoding systems. In this framework, meta-reporting capabilities are considered an important criterion for differentiating geocoding systems. For example, the spatial area of reference features was used to approximate uncertainties and select multiple candidates for the geocoding process [26]. Researchers have pointed out that better quantifications for geocoding/reverse geocoding workflow (i.e., from input to output) could benefit subsequent studies that employ the geocoded data [117, 43].

Our work extends the existing works in two main aspects. First, the proposed workflow and algorithms purely utilized the spatial attributes, geometries, and typologies of existing address models to improve the accuracy of output data without other supplementary information. Second, we explicitly quantify uncertainty for the entire reverse geocoding workflow based on the likeli-

hood of candidates being the best and incorporated input GPS uncertainties to generate the final candidate list as well as corresponding metadata descriptions.

4.3 Workflow Formalization

Generally, a reverse geocoding workflow can be summarized as the finding and ordering the nearest address descriptions using certain criteria, given an input latitude and longitude. Figure 4.1 describes the proposed workflow. It consists of three main components: Reference Datasets, K-Nearest Search, and Spatial Topology Validations.

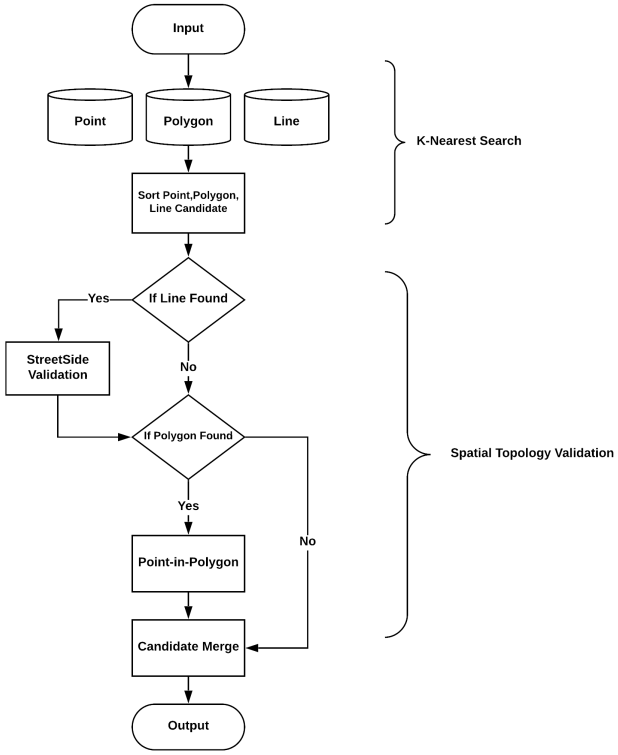


Figure 4.1: Proposed reverse geocoding workflow

4.3.1 Reference Datasets

All three address models are utilized to leverage the geometrical and topological characteristics for different aspects of the workflow. Specifically, point and polygon models are used to derive

output candidates considering their typical high spatial accuracy. Line models are used to identify the nearest streets for spatial topology validations.

4.3.2 K-Nearest Search

To retrieve nearby candidates given a GPS input, the shortest distances from the input to the reference data need to be calculated. In this work, we adopt the Euclidean equation between coordinates of two points under the same projection (i.e., WGS84) for distance calculation as follows.

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4.1}$$

This distance calculation has the following three advantages: (1) possible candidates should be distributed near input coordinates, so this nearby area could be considered as a planar without affecting the accuracy of distance calculation [30]; (2) the purpose of distance calculations is to assign each candidate a ranked order rather than obtaining precise distance values; (3) the calculation of Euclidean distance between two coordinates is less computationally intensive than other measures such as great circle distance. As shown in Figure 4.2, we specify the distance calculation based on the geometry of the reference data : point, line, polygon.

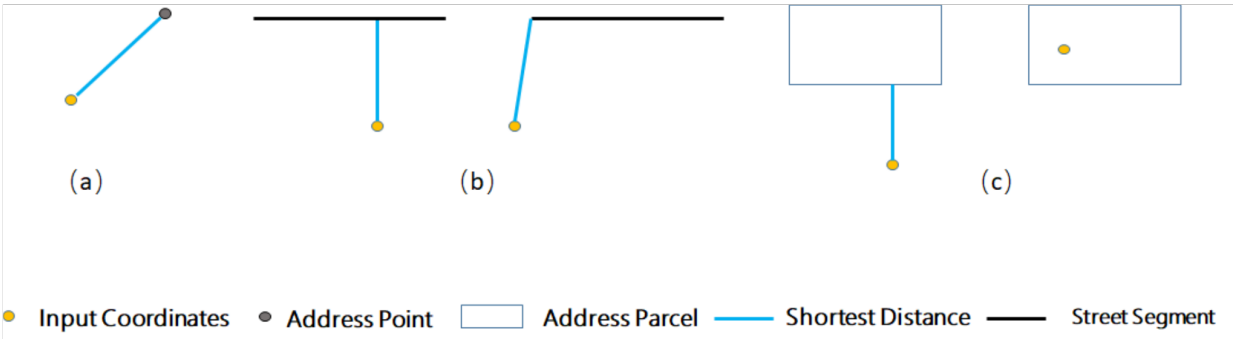


Figure 4.2: Shortest distance from input coordinates to different address models: (a) Address Point. (b) Street Segment. (c) Address Parcel.

Point: The shortest distance D_{point} from the pair of input coordinates to a point reference data

can be directly calculated using Equation 4.1.

Line: As shown in Figure 4.2 (b), the shortest distance D_{Ls} between a street segment Ls and input coordinates has two cases, depending on the intersection point of the perpendicular line from input coordinates to the street segment, $P_{intersect}$, is on this street segment or not. We calculate D_{Ls} using Equation 4.2, where $D_{intersect}$, D_{start}, and D_{end} denotes the Euclidean distance from input coordinates to $P_{intersect}$ and start/end-point of street segments, respectively.

$$D_{Ls} = \begin{cases} D_{intersect} & \text{If } P_{intersect} \text{ on } Ls \\ \min(D_{start}, D_{end}) & \text{Otherwise} \end{cases} \tag{4.2}$$

If multiple street segments exist in a street, we iterate through each segment Ls_i of the whole street St to obtain the global minimal distance value as the shortest distance D_{line} as follows:

$$D_{line} = min\{D_{Ls_i} \mid Ls_i \in St\}, \tag{4.3}$$

Polygon: As depicted in Equation 4.4, if the input point I is inside a polygon, we can always find a point inside this polygon which overlaps with I. As a result, the shortest distance $D_{polygon}$ will be zero.

$$\min_{p \in P_{polygon}} D_{polygon}(p, I) = ||I - p||_2^2 = 0; \tag{4.4}$$

If I is outside a polygon, $D_{polygon}$ should stretch from I to one edge of this polygon. Because polygon edges are identical to line segments, we iterate through each edge of this polygon and use Equation 4.2 and 4.3 to obtain $D_{polygon}$ as follows:

$$D_{polygon} = min\{D_{E_i} \mid E_i \in Polygon\} \tag{4.5}$$

To this end, $D_{polygon}$ can be determined as:

$$D_{polygon} = \begin{cases} 0 & \text{If the input point inside the polygon} \\ min\{D_{E_i} \mid E_i \in Polygon\} & \text{Otherwise} \end{cases} \tag{4.6}$$

4.3.3 Spatial Topology Validation

We propose a series of topological relationships to further validate candidates, inspired by the usage of topology relations for candidate selections in standard geocoding processes [26]. Given the geometry of the input point and possible address candidates (i.e., point, line, and polygon), we utilize three spatial topological relationships: *Containment*, *Intersection*, and *Disjoint*, which are drawn from the 4 - intersection model [118]. It is important to identify a correct nearby street from all retrieved line data to conduct spatial topology validation. There are two reasons that the street with the nearest distance to the input is not always the most appropriate: (1) the true positive nearby street may not exist in the reference datasets, and (2) the location description for an input and the nearest street may have different descriptions, especially for buildings are located near the end of street blocks. Therefore, we check whether the street descriptions of each candidate agrees with the retrieved nearest line data and determine the most appropriate candidate in a quorum manner. Namely, we first check if there is a street candidate with the same description as the first point candidate. If not, we then attempt to find a street candidate that agrees with the first polygon candidate. If still unsuccessful, we choose the street candidate with the description that coincides with the majority (i.e., larger than 50%) of point and polygon candidates. If all failed, we select the nearest street candidate.

4.4 Weight-based Quantification for Reverse Geocoding

To rank multiple candidates consistently, we quantify the proposed workflow (i.e., *K-Nearest Search* and *Spatial Topology Validation*) by a weight that indicates how likely one candidate is to be the best.

Table 4.1: Spatial topology validation rules

Retrieved Reference Data	Spatial Relations	Descriptions
Point data, Polygon data, Line data	Intersection	Determine points and polygon candidates that are located on the same side of a street as the input if the connected lines between candidates and the input point do not intersect with the street line.
Polygon data, Point data	Containment	Determine the containing polygon. Namely, a polygon candidate only contains the input point or contains the input point and any other candidates.
Point data, Polygon data	Containment	Determine inside candidates, namely candidates are inside the containing polygon.
Point data, Polygon data	Disjoint	Determine point or polygon candidates that are outside the containing polygon

4.4.1 Quantification for K- Nearest Search

In the K-Nearest Search phase, the distance between a candidate in reference datasets and input coordinates is a key metric for discriminating among candidates. This metric has been used to quantify the weight of linear interpolation for (standard) geocoding processes [119]. Thus, we consider the distance D from candidates to input coordinates as an approximation of the weight W that one candidate is the best. Ideally, W decreases monotonically with D and must meet:

$$\begin{cases} \lim_{D\to\infty} f(D) = 0 \\ \lim_{D\to 0} f(D) = 1 \end{cases} \tag{4.7}$$

Thus, we define the weighting function as the following:

$$W = \exp(-D) \tag{4.8}$$

With the substitution of D with the shortest distance obtained from Equation 4.1 to Equation 4.6, we can quantify the candidates retrieved from the K-Nearest Search process using the same

standard.

4.4.2 Quantification for Spatial Topology Validation

Our spatial topology quantification approach derives weights for two scenarios: (1) any polygon candidates that contain the input coordinates, and (2) any point or polygon candidates that are located on the same side or opposite side of the street as that of the input coordinates.

Polygon Containment Validation: The weight calculation for polygon candidates is specific to building distributions. In most cases, such as residential housing areas, the polygon candidate only contains the input coordinates. Such a polygon receives a full score of 1. The weight of other candidates that do not contain the input coordinates is proportional to the distance d' to input coordinates based on Equation 4.8. However, in some corner cases such as multiple buildings inside a shopping area, as depicted in Figure 4.3 (a), we can find a polygon that contains multiple points or polygon candidates (i.e., buildings), denoted as "containing polygon". Intuitively, buildings inside this area (i.e., address *201 -204*) should receive higher weights than the area itself (i.e., address *200*) and other outside buildings (i.e., address *210*). To do so, we first obtain the maximum distance (denoted as d_{max}) between input coordinates and each inside candidate. We then apply $exp(-(d_{max}+1))$ to ensure the containing polygon receives a lower weight than that of any inside candidates. To ensure outside candidates receive lower weights than that of the containing polygon, their weights are calculated as $exp(-(d_{max}+d'))$, where d' denotes their distance to the input coordinates. To this end, we have

$$W = \begin{cases} 1 & \text{Containing polygon \& if containing polygon only contains input} \\ exp(-d') & \text{Outside candidates \& if containing polygon only contains input} \\ exp(-(d_{max}+1)) & \text{Containing polygon \& if this polygon contains input and other candidates} \\ exp(-(d_{max}+d'+1)) & \text{Outside candidates \& if this polygon contains input and other candidates} \\ exp(-d') & \text{Inside candidates \& if containing polygon contains input and other candidates} \end{cases} \tag{4.9}$$

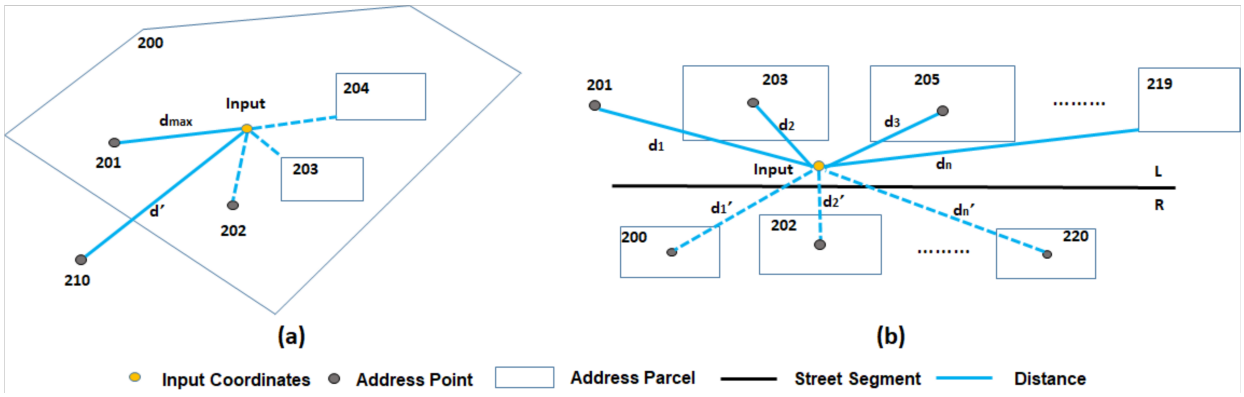


Figure 4.3: Corner cases for topology validation. (a) A large area contains multiple buildings; (b) Opposite-side buildings is extremely close to input coordinates.

Street Side Validation: To boost candidates that are located on the same side as that of input coordinates, we define the following equation:

$$W = \begin{cases} 1 & \text{If candidates located on the same side as input} \\ exp(-(d_{n-i+1} - Min(d'_1, ..., d'_n) + 1 + d')) & \text{Otherwise} \end{cases} \tag{4.10}$$

As shown in Figure 4.3 (b), we assign a full score of 1 to each candidate located on the same side (i.e., 201-219). In contrast, candidates located on the opposite side (i.e., 200-220) have weights that are proportional to their distance to the input point. To calculate weights for opposite-side candidate, we introduce a parameter *i* to indicate the position where the nearest opposite-side candidate could appear. For example, in Figure 4.3 (b), if *i* equals to 6, the nearest opposite-side candidate 202 will be ranked sixth in the final output. We descendingly order the distance to the input point for the same-side candidates (d_1 to d_n) and the opposite-side candidates (d'_1 to d'_n). The weight of each opposite-side candidate is calculated as $exp(-(d_{n-i+1} - Min(d'_1, ..., d'_n) + 1 + d'))$, where d' is the distance from input coordinates to this candidate, and *n* is the number of candidates. To this end, candidates on the opposite-side always receive lower weights than those on the same-side, and the rank of the nearest opposite-side candidate is configurable by end-users.

4.4.3 Quantification for Candidate Fusion

In short, the final weight for each candidate is accumulated through each of the quantification processes in our workflow as follows:

$$W_{final} = \prod W_{process} \tag{4.11}$$

As indicated in Section 4.4.1, nearest candidates and their corresponding weights will be generated from the K-Nearest Search process. We group each pair of address descriptions and their weights into three sets based on the geometry of retrieved candidates, denoted as $Dict_{point}$, $Dict_{polygon}$, and $Dict_{line}$, respectively. Next, we use the proposed spatial topology rules to validate each candidate and obtain a new weight. Because point and polygon candidates come from separated reference sources, identical address descriptions may exist in both point and polygon candidates sets. We merge candidates based on their address descriptions to generate final output candidates. Since the reference datasets used were processed by [26], we conduct exact comparisons of every address component value to determine the duplicate address descriptions. Finally, we calculate the final weight for each candidate as follows.

$$W_{Addr} = \max(W_{point}, W_{polygon}) \tag{4.12}$$

To this end, $Dict_{point}$ and $Dict_{polygon}$ are merged into a new set $Dict_{result}$, which contains candidates with unique address descriptions and their corresponding weights. Naturally, the top candidates should share the same street name (i.e., addresses distributed on two sides of a street). Thus, we track whether the first candidate street name agrees with that of the other candidates. If not, we apply a downgrading factor of 0.9 for them to uplift the rankings of candidates that have identical street names.

Algorithm 5: Weight-based Candidate Ranking (WCR)

Input : Input GPS Point pnt; $Dict_{polygon}$, $Dict_{point}$, $Dict_{line}$ from K-Nearest Search; the position of nearest opposite-side candidate i

Output: A set of Ranked Addresses $Dict_{result}$

Function *WCR (pnt, i)* **:**
- **if** $Dict_{polygon}$ *exists* **then**
  - $Dict_{point}$, $Dict_{polygon}$ = Polygon Containment Validation (pnt,$Dict_{point}$, $Dict_{polygon}$);
- **end**
- **if** $Dict_{line}$ *exists* **then**
  - $St = Quorum(Dict_{line})$;
  - $Dict_{point}$, $Dict_{polygon}$ = Street Side Validation (pnt,$Dict_{point}$, $Dict_{polygon}$, St, i);
- **end**
- $Dict_{result}$ = Merge $Dict_{point}$, $Dict_{polygon}$ using Equation 4.12;
- **Apply** 0.9 * W_{Addr} in $Dict_{result}$ for an address description differs from the street name;
- **Sort by** W_{Addr} in $Dict_{result}$;
- **Return** $Dict_{result}$;

Function *Street Side Validation (pnt,*$Dict_{point}$*,* $Dict_{polygon}$*, St, i)* **:**
- $sameSideDistArr = [\,]$, $oppoSideDistArr = [\,]$;
- **for** $cand$ *in* $Dict_{point}$ *and* $Dict_{polygon}$ **do**
  - L_{pnt} = Connect pnt to $cand$;
  - **if** L_{pnt} *intersects with* St **then**
    - $oppoSideDistArr.add(d_{cand})$;
    - $W_{cand} = exp(-d_{cand})$
  - **end**
  - **else**
    - $sameSideDistArr.add(d_{cand})$;
    - $W_{cand} = 1$
  - **end**
- **end**
- **Sort** $sameSideDistArr$ descendingly;
- $d_{pe} = sameSideDistArr[n - i + 1] - Min(oppoSideDistArr) + 1$;
- **for** $cand$ *in* $Dict_{point}$ *and* $Dict_{polygon}$ **do**
  - **if** $W_{cand}! = 1$ **then**
    - $W_{cand} *= exp(-d_{pe})$
  - **end**
- **end**
- **Return** $Dict_{point}$, $Dict_{polygon}$;

Function *Polygon Containment Validation (pnt,* $Dict_{point}$*,* $Dict_{polygon}$*)* **:**
- **for** $polyg$ *in* $Dict_{polygon}$ **do**
  - A_{polyg} = the area of $polyg$;
  - **if** pnt *inside* $polyg$ *&* A_{polyg} *is minimal* **then**
    - $containPolygon = polyg$;
    - $W_{containPolygon} = 1$;
  - **end**
- **end**
- **if** *containPolyg exits* **then**
  - $distArr = [\,]$, $outsideCandArr = [\,]$;
  - **for** $cand$ *in* $Dict_{point}$ *and* $Dict_{polygon}$ **do**
    - **if** $cand$ *Inside* $containPolygon$ **then**
      - $distArr.add(d_{cand})$;
      - $outsideCandArr.add(cand)$;
    - **end**
  - **end**
  - **if** *Length of* $distArr! = 0$ **then**
    - $d_{max} = Max(value \in distArr)$;
    - $W_{containPolygon} = exp(-(d_{max} + 1))$;
    - **for** $cand \in outsideCandArr$ **do**
      - $W_{cand} *= exp(-d_{max})$
    - **end**
  - **end**
- **end**
- **Return** $Dict_{point}$, $Dict_{polygon}$;

4.5 Input Uncertainty Propagation

To propagate GPS uncertainty into the final outputs, we first derive a statistical surface for GPS accuracy and then incorporate this uncertainty into final candidate ranking lists.

4.5.1 Input GPS Uncertainty Statistical Surface

On Android-based phones, the accuracy of a GPS signal is described by the Circular Error Probability (CEP) circle, and there is a 68% probability that a true GPS location will be located within that circle [120]. For example, if a GPS reads "19 meters," there is a 68% probability that the true location is inside the circle with a radius of 19 meters. A previous study suggested that the distribution of the true GPS location inside a CEP circle is not uniform, and that the distribution of error distances between the true GPS location and the input coordinates can be well approximated by the Rayleigh distribution [121]. Based on their findings, we have the cumulative distribution function of the Rayleigh distribution as:

$$F = 1 - \exp(-\frac{R^2}{2\sigma^2}) \tag{4.13}$$

where R is the radius of a CEP circle, and F is the probability that the true GPS is inside this circle. Therefore, we can obtain the scalar parameter σ using Equation 4.14.

$$\sigma = \sqrt{\frac{R^2}{-2\ln(1-F)}} \tag{4.14}$$

In this work, the maximum radius we consider is the radius of a circle that has a 95% probability of containing the true GPS position, denoted as R_{95}. We do not include the remaining 5%, because these points are too far away from the input according to this distribution. Thus, we can rewrite the cumulative distribution function as:

$$\int_0^{R_{95}} \frac{R}{\sigma^2} \exp(-\frac{R^2}{2\sigma^2})\, dR = 0.95 \tag{4.15}$$

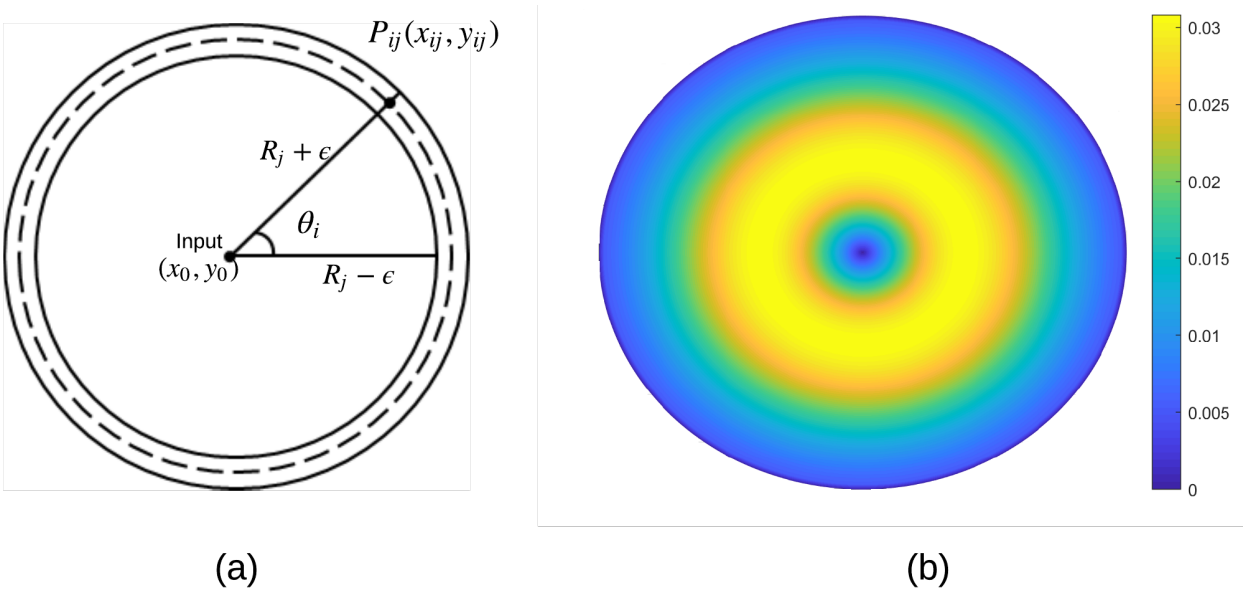


Figure 4.4: GPS circular error probability.

Given Equation 4.15, CEP is the integral of circles with different radii. Each $Cir \in$ CEP has its probability with respect to radius R, and each Cir comprises a set of possible GPS locations, denoted as $p \in Cir$. Figure 4.4 (a) depicts a particular circle that composes the whole CEP and a possible GPS location p_{ij} (x_{ij}, y_{ij}) on this circle, given input coordinates (x_0, y_0). To obtain the probability that the true GPS point appears at any positions in the CEP, we have to first obtain the probability of a Cir with a particular R. However, due to the properties of continuous probability distributions, we could not directly obtain the probability of each Cir by Equation 4.15. Therefore, we introduce the parameter ϵ to represent the small offset distance for any radius R_j. Therefore, we can approximate the probability of Cir with $R_j \in [0, R_{95}]$ as:

$$Pr_{R_j} = \int_{R-\epsilon}^{R+\epsilon} \frac{R}{\sigma^2} \exp(-\frac{R^2}{2\sigma^2})\, dR = \exp(-\frac{(R-\epsilon)^2}{2\sigma^2}) - \exp(-\frac{(R+\epsilon)^2}{2\sigma^2}) \tag{4.16}$$

In this work, we set ϵ as $\frac{1}{2000}$ meters, which is exceedingly small in comparison to CEP radii. To indicate different positions of points on a Cir, we introduce the other parameter θ as the bearing related to the horizontal line. We assume it has a uniform distribution. To this end, the position of a GPS point can be described as a position that has θ_i related to the horizontal line and has a

distance of R_j to the input coordinates, denoted as p_{ij}. We define the increment for θ and R as $\frac{\pi}{18}$ degrees and $\frac{R}{100}$, respectively. Since the total probability is 0.95, we normalize the probability of p_{ij} as:

$$Pr_{p_{ij}} = \frac{Pr_{R_j}}{36 \sum_{0}^{R_{95}} Pr_{R_j}} \times 0.95 \tag{4.17}$$

where Pr_{R_j} is the probability obtained by Equation 4.16. Finally, we calculate coordinates of this point p_{ij} using the distance R_j and the bearing θ_i related to the input GPS point as follows:

$$\begin{cases} x_{ij} = x_0 + R_j \cos\theta_i \\ y_{ij} = y_0 + R_j \sin\theta_i \end{cases} \tag{4.18}$$

In the end, the point p_{ij} is paired with a probability $Pr_{p_{ij}}$ for the coordinates (x_{ij}, y_{ij}) to be the true GPS location. Figure 4.4 (b) shows the statistical surface of points appearing at different positions in a CEP circle with a radius of 25 meters, which is analog to previous findings for how GPS points clustered for a fixed position [113, 122].

4.5.2 Input GPS Uncertainty Propagation

To incorporate input GPS uncertainty into final candidate lists, the proposed reverse geocoding workflow can be considered a function of any input coordinates (x_{ij}, y_{ij}). Given an input, the reverse geocoding process will output a set of address candidates $A = \{Addr_1, Addr_2, \cdots, Addr_n\}$. Each $Addr_i \in A$ is ranked by a calculated $W_{AddressRecord}$, which is the likelihood of a candidate to be the best. Meanwhile, the input point (x_{ij}, y_{ij}) has a probability $Pr_{p_{ij}}$ of being the true GPS location. Since the reverse geocoding process and the true GPS location are independent, we propagate input uncertainties into and through the proposed workflow using the conditional probability production rule. In the end, the ranked candidates returned by the proposed workflow can be considered to have uncertainty values propagated end to end, from input to output.

Algorithm 6: Input GPS Uncertainty Incorporation

Input : Input GPS Coordinates (x_0, y_0); GPS CEP Radius R_{phone}
Output: A set of Ranked Addresses $Dict_{result}$

$I \leftarrow$ Input GPS Coordinates (x_0, y_0);
$R_{95} \leftarrow$ GPS CEP Radius R_{phone};
$Dict_{result} \leftarrow \{\}$;
for $\theta_i = \theta_i; \theta_i <= 2\pi; \theta_i = \theta_i + \frac{\pi}{18}$ **do**
    for $R_j = R_0; R_j <= R_{95}; R_j = R_j + \frac{R_{95}}{100}$ **do**
        $Dict_i \leftarrow \{\}$;
        $P_{p_{ij}} \leftarrow \frac{Pr_{R_j}}{36 \sum_0^{R95} Pr_{R_j}} \times 0.95$;
        $x_{ij} \leftarrow x_0 + R_j \cos\theta_i$;
        $y_{ij} \leftarrow y_0 + R_j \sin\theta_i$;
        $Dict_i \leftarrow WCR(x_{ij}, y_{ij})$;
    end
    for $Addr_j \in Dict_i$ **do**
        $W_{Addr_j} \leftarrow W_{Addr_j} * P_{p_{ij}}$;
        $Dict_{result} \leftarrow Dict_i$;
    end
end
Sort by W_{Addr} in $Dict_{result}$;
Return $Dict_{result}$;

4.6 Experiments and Results

4.6.1 Experimental Setup

We evaluated the effectiveness of the proposed approach by answering three questions: (1) Does the proposed workflow deliver more accurate first candidates; (2) Does the proposed approach efficiently rank candidates that strongly align with the ground-truth ranks; and (3) Does consideration of input GPS uncertainty benefit final output candidates.

Reference data used include address point, address parcel, and street line. Address point data were extracted from Navteq 2016 dataset[6]. Address parcel data were obtained from National Boundary Solution databases. These two datasets are commercial datasets. Street line data were

[6] https://www.here.com/en/navteq

generated from the US Census Bureau TIGER/Lines[7], which is an open-source dataset. The testing dataset was building rooftop centroid coordinates, obtained from the city of College Station, TX. Each pair of coordinates was associated with an address description on one of three land-use types: Low-density Residential (LDR), High-density Residential (HDR), and Commercial (COM).

To conduct the evaluation, the proposed approach was implemented as a web service using Node.js[8]. Reference data were indexed in the Elasticsearch[9] NoSQL database. Visualization was implemented using Leaflet[10].

The first and second experiments assume that input points are accurate without uncertainty. To simulate people using reverse geocoding services inside buildings, we directly used building centroids as inputs. To simulate people using reverse geocoding services outside buildings, we employed the street geocoding techniques of Texas A&M GeoServices[11] to "place" these centroids on streets. It is worth noting that street geocoding and the proposed approach are two independent systems. Thus, the generated on-street coordinates have no correlation with the output from reverse geocoding. For the third experiment, we collected input points with various degrees of errors using an Android-based phone.

4.6.2 Results and Discussions

4.6.2.1 Correctness of the First Candidate

To answer the first question, we compared the first candidate produced by the proposed approach-Weighted based Candidate Ranking (denoted as WCR) to the open-access Texas A&M GeoServices (denoted as TAMU), which uses the same reference data as the proposed approach. We also compared WCR with four other commercial reverse geocoding systems: Google Maps, Bing Maps, Esri, and Here WeGo. To better capture the characteristics of building distributions across locations, we randomly selected one address on every street from the test dataset. In total, we used 1616 building centroid and 1,477 valid on-street coordinates. We ran this sample through each

[7] https://www.census.gov/geo/maps-data/data/tiger-line.html
[8] https://nodejs.org
[9] https://www.elastic.co/products/elasticsearch
[10] https://leafletjs.com/
[11] http://geoservices.tamu.edu/

system and summarized the correctness rate produced by each system. Table 4.2 shows that our approach achieved the fourth and third rankings among the six systems, in terms of correctness rate, under the indoor and outdoor scenarios, respectively. It is important to note that commercial systems may have better reference data, which are not revealed publicly [46, 26]. Therefore, a comparison of our approach to the system that uses the same reference dataset can better reveal the advantage of the proposed workflow. Compared to TAMU, which uses the same reference data, the proposed approach improved the correctness rate by 4.02% and 6.58% under the indoor and outdoor scenarios, respectively.

Table 4.2: Correctness of first candidates

Scenario	WCR	TAMU	Google	Microsoft	Esri	Here
Indoor	83.35%	79.33%	93.44%	89.85%	89.36%	81.06%
Outdoor	84.63%	78.05%	81.72%	87.41%	89.38%	76.1%

To further explore the performance of each system, we manually reviewed all outputs produced by each system and classified unreasonable outputs into four error categories. These four categories are similar to previous (reverse) geocoding quality assessments: (I) Spatially far away from the input coordinates (i.e., incorrect house numbers or street name). (II) Reverted to an address range (i.e., correct street name). (III) Opposite side of the street (i.e., opposite-side house number and correct street name). (IV) Reverted to a large geographic area (i.e., cities or postal areas). Figure 4.5 summarizes the unreasonable outputs produced by each system. According to this figure, our approach mainly had Error I. The reason for this phenomenon is the coverage and quality of the reference data. As shown in the examples in Figure 4.6 (a), our approach only retrieved a large polygon representing the whole building complex rather than particular buildings, resulting in the first candidate being far away from the input point (i.e., Error I). The limited coverage of our

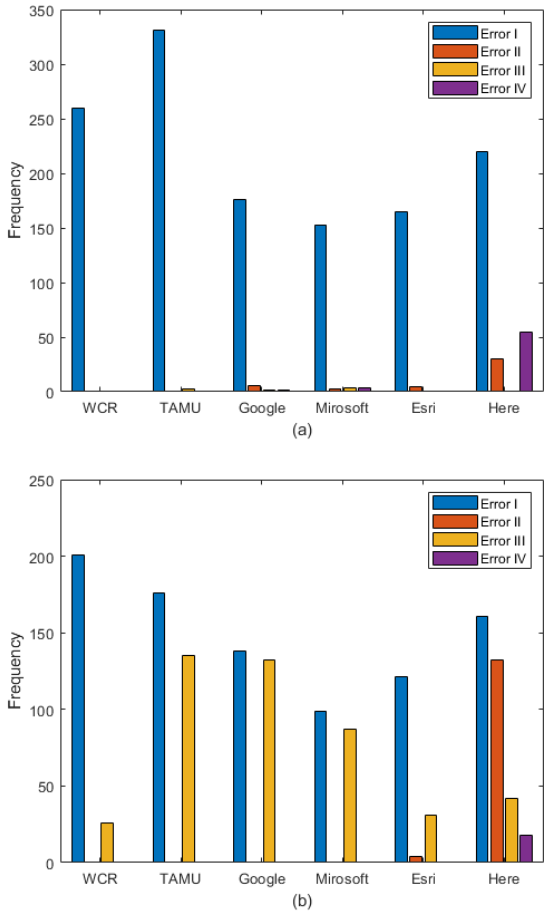
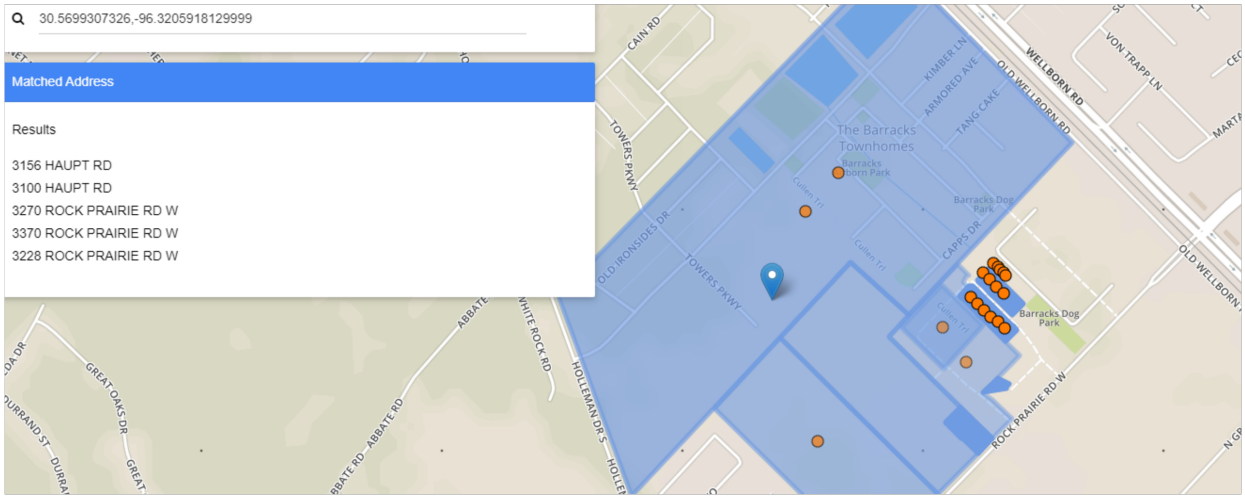
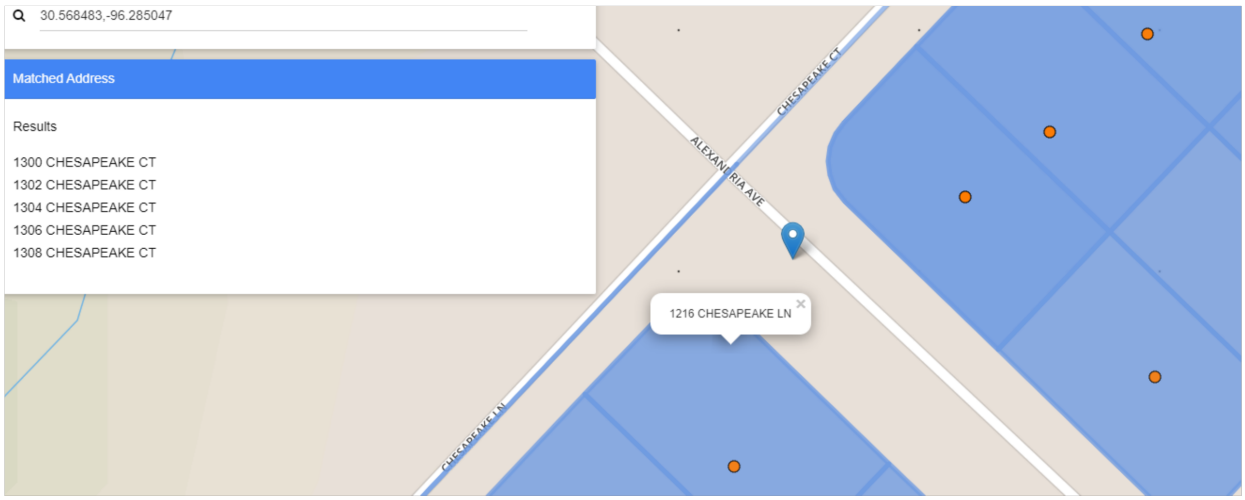


Figure 4.5: Frequency of error categories under indoor (a) and outdoor (b) scenarios

reference dataset also resulted in Error III. As shown in Figure 4.6 (b), because the correct nearby street did not exist in our reference datasets, we could not filter out the opposite-side candidates. Likewise, TAMU has a large extent of Error I due to the same reference data used.

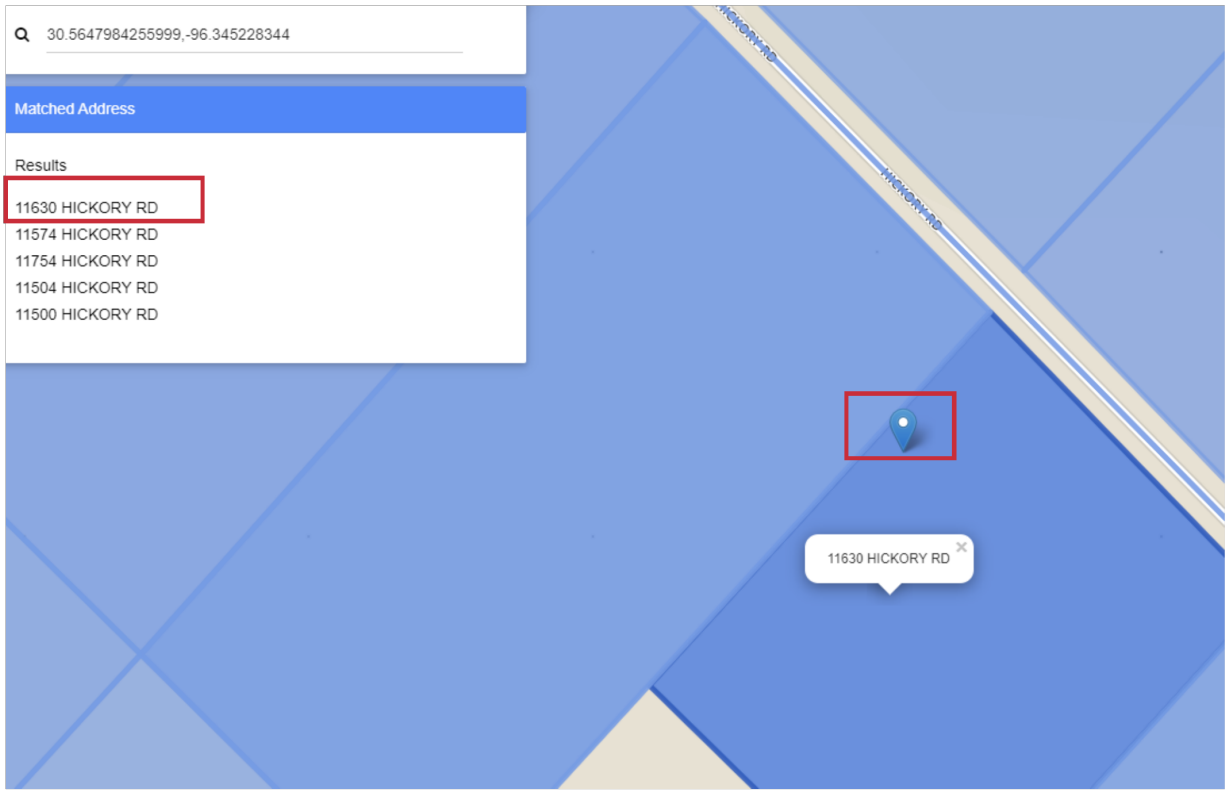


(a)

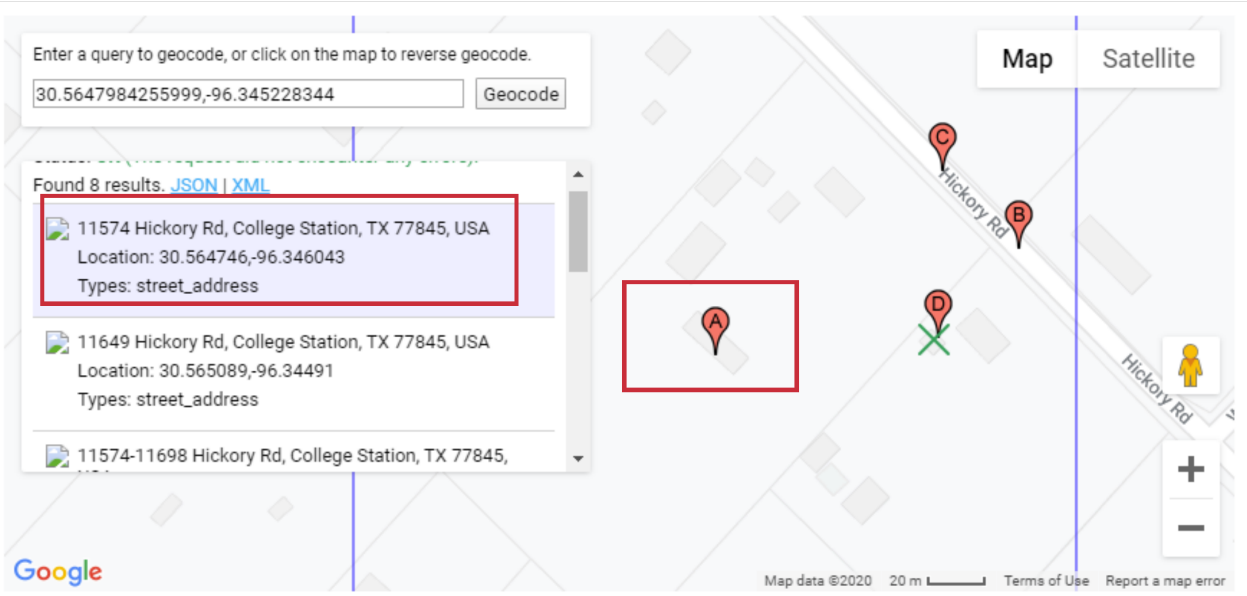


(b)

Figure 4.6: Impact of missing reference data on candidate selection

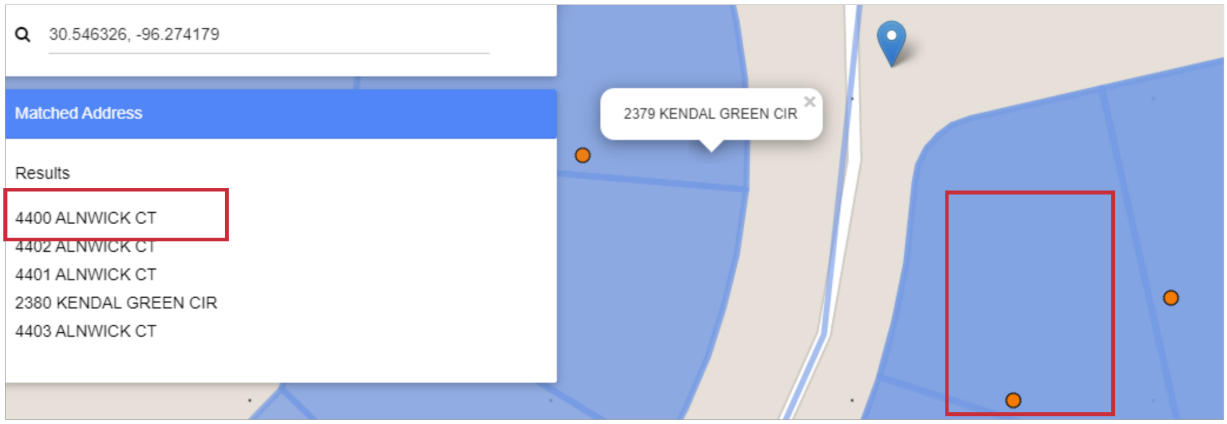


(a)

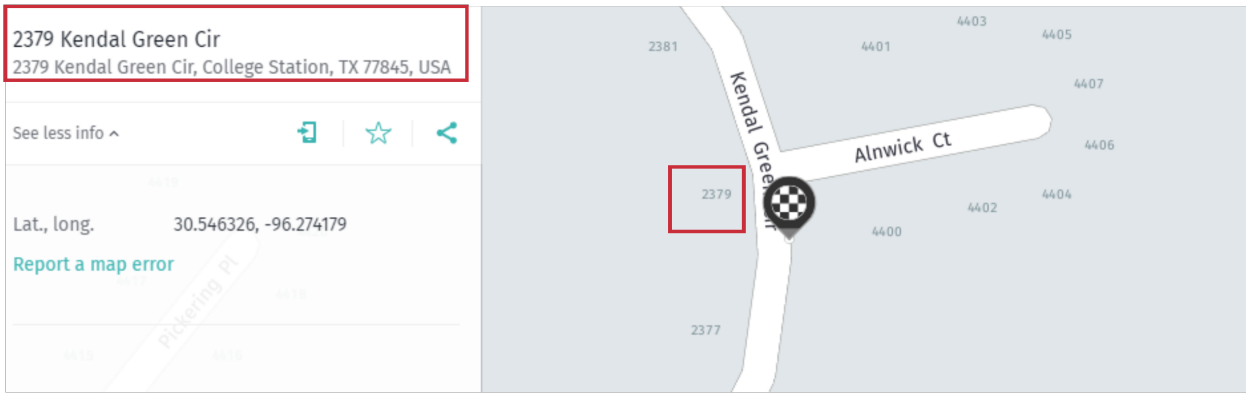


(b)

Figure 4.7: First candidate for our approach (a) and Google (b) for indoor coordinate



(a)



(b)

Figure 4.8: First candidate for our approach (a) and Here (b) for outdoor coordinate

In terms of input scenarios, inputs in indoor scenarios are building centroids, which are easier for each system to determine the nearest candidates. In most cases, if a parcel only contains the input point, our approach primarily uses the containment relationship to generate output candidates. If multiple candidates are also inside this parcel, our approach determines candidates using Equation 4.9. Even though indoor inputs are relatively straightforward, we still observed the advantage of conducting spatial topology validation. As shown in Figure 4.7 (b), Google returned a Error I candidate: *11574 Hickory Rd*, which corresponds to a parcel next to the input. Conversely, our approach (Figure 4.7 (a)) utilized the *containment* relationship to output the correct candidate. Compared to indoor scenarios, inputs in outdoor scenarios involve more topological relationships,

and the benefit of spatial topology validations for candidate selection becomes more significant as input points move toward street center-lines. Figure 4.8 shows that once the nearby street was found, we could avoid outputting candidates located on the opposite side of the street. In contrast, other reverse geocoding systems may not conduct this process, resulting in a larger extent of Error III in both scenarios. These findings help explain that our approach had fewer Error III incidences and had a higher ranking of correctness rate in outdoor scenarios than that in indoor scenarios, as shown in Table 4.2. Errors II and IV can be explained by the search radius and logic used by each commercial vendor. We found that Here tends to conduct linear interpolations at the street level to generate the final output. It is interesting to note that 366 outputs had spatially correct locations but also had address descriptions that differed across other commercial providers. This finding is particularly important because the usage of such linear interpolation approaches could result in address descriptions, different from those produced by other systems, creating additional communication difficulties. In contrast, our approach and TAMU avoided Error II by limiting the usage of linear interpolation on the street level. Arguably, the limited knowledge of reference data and search logic used by commercial systems prevent us from drawing concrete conclusions for the advantage of our approach over these commercial systems. The improvement of correctness rate in comparison to the system that uses the same reference data and in a more complicated input scenario shows the validity of the proposed workflow.

4.6.2.2 Agreement of Candidate Ranking

To answer the second question, we compared the top five candidates obtained from our system to those ranked by humans and examined the level of their agreement. To avoid the impact of missing reference data on ranking results, we randomly selected input coordinates that could be resolved to the correct address descriptions from the first experiment. In total, we collected 130 building-centroid coordinates and 130 on-street coordinates on three land-use types (i.e., 50 LDR, 30 HDR, and 50 COM). We recruited 10 reviewers to rank the top five candidates for each input. Each reviewer was assigned 13 building-centroid and 13 on-street coordinates, which were randomly selected from these three land-use types. Then, we used between-subjects repeated mea-

sure design to address possible disagreements on the ranking of these candidates [123, 124]. In particular, for each reviewer, we randomly switched their judge results to other reviewers, making sure each ranking was repeatedly judged by other reviewers. The study resulted in a 15% disparity in the judgments. We found that this is because the reviewers were not sure if the fifth candidate should be located on the same-side or the opposite-side. Finally, for the small percentage of disparity, we discussed the disagreements and considered driving or walking accessibility and made the final agreements on the rankings. We employed *normalized Discounted Cumulative Gain* (nDCG) to quantify the agreement between system output and human judgment. This metric is widely used to evaluate output quality for search engines and POI suggestions [125, 106]. We chose nDCG over other common metrics such as Mean Reciprocal Rank (MRR) because we already evaluated the quality of the first candidates and wanted to focus on the ranking quality of the remaining positions. In this work, nDCG@K calculated by:

$$nDCG@K = \frac{DCG@K}{IDCG@K} \tag{4.19}$$

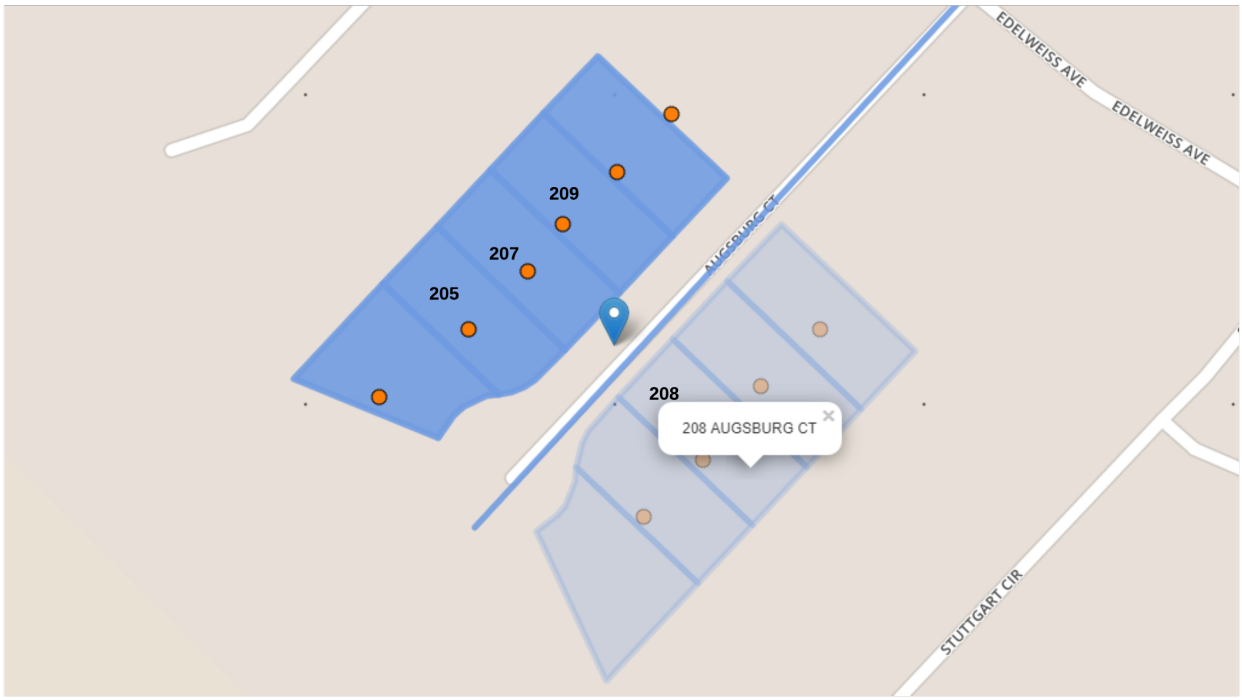
where $DCG@K = \sum_{i=0}^{K} \frac{Score_i}{log_2(i+1)}$ and $IDCG@K = \sum_{i=0}^{K} \frac{Score'_i}{log_2(i+1)}$. $Score'_i$ represents the ideal relevant score for each candidate. This score decreases from 5 to 1 as its position i in the human-judged ranking increases from 1 to 5. $Score_i$ is the relevant score assigned to a candidate according to its ranking i in human-judged rankings. If a candidate does not exist in the human-judged ranking, it receives a score of 0. We specified the values of K to be 3 and 5, as these are common numbers of candidate suggestions. We made comparisons to Here and Google, because only these two commercial systems allow returning multiple candidates through their API endpoints. To avoid the impact of reference data on rankings, we compared our approach to a distance-only ranking method that uses the same reference data, denoted as Distance. We also compared the ranking results obtained by our approach with different settings for the position in which the nearest opposite-side candidate could appear, denoted as WCR′i. Table 4.3 summarizes nDCG@3 and nDCG@5 scores for different ranking approaches. The comparison of nDCG@3 under WCR′2

Table 4.3: nDCG@K Comparisons

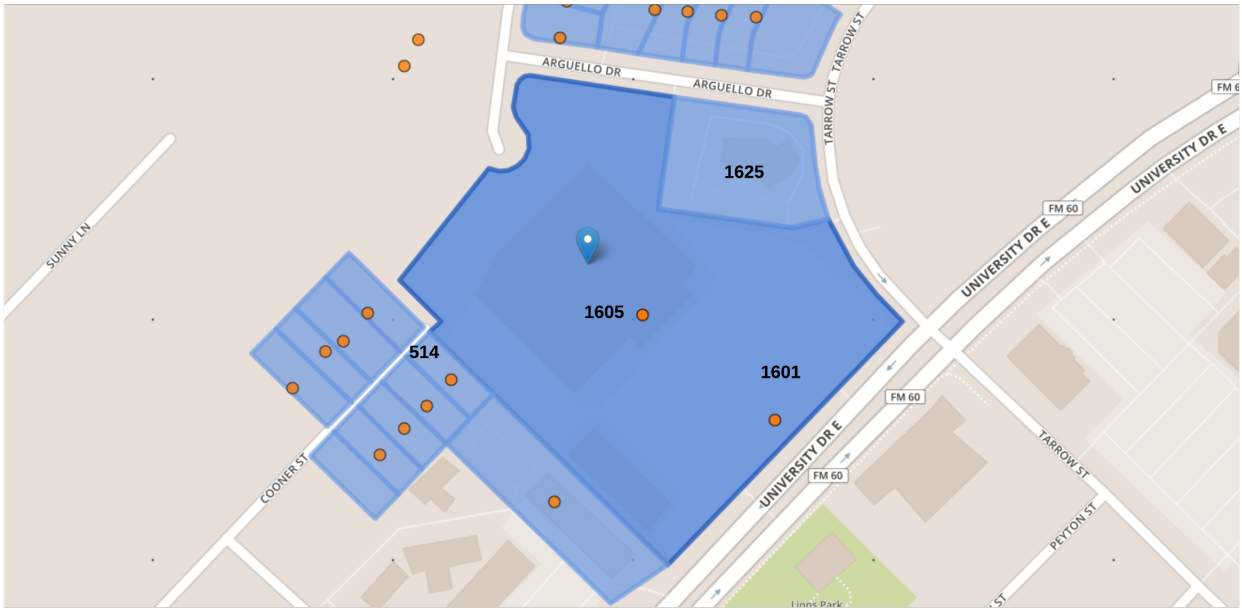
Input Scenarios	Method	nDCG@3				nDCG@5			
		LDR	HDR	COM	Total	LDR	HDR	COM	Total
Indoor	WCR′2	0.7188	0.7435	0.8100	0.7596	0.7727	0.7891	0.8125	0.7918
Indoor	WCR′3	0.8750	0.8597	0.8604	0.8659	0.8576	0.8415	0.8428	0.8482
Indoor	WCR′4	0.9626	0.9169	0.8970	0.9268	0.8922	0.8804	0.8734	0.8822
Indoor	WCR′5	0.9626	0.9169	0.8970	0.9268	0.9278	0.8961	0.8896	0.9058
Indoor	WCR′6	0.9626	0.9169	0.8970	**0.9268**	0.9376	0.9011	0.8934	**0.9122**
Indoor	WCR′7	0.9626	0.9169	0.8970	0.9268	0.9376	0.9011	0.8934	0.9122
Indoor	Distance	0.9453	0.8643	0.8599	0.8937	0.9120	0.8539	0.8393	0.8706
Indoor	Google	0.6322	0.5894	0.5531	0.5919	0.5622	0.5189	0.4746	0.5185
Indoor	Here	0.5737	0.5632	0.5086	0.5462	0.5123	0.5057	0.4839	0.4999
Outdoor	WCR′2	0.7350	0.8034	0.7596	0.7602	0.7827	0.8287	0.7636	0.7860
Outdoor	WCR′3	0.8535	0.8967	0.8345	0.8561	0.8543	0.8781	0.8148	0.8446
Outdoor	WCR′4	0.9576	0.9503	0.8932	0.9312	0.8988	0.9198	0.8578	0.8879
Outdoor	WCR′5	0.9576	0.9503	0.8932	0.9312	0.9347	0.9334	0.8801	0.9134
Outdoor	WCR′6	0.9576	0.9503	0.8932	**0.9312**	0.949	0.9347	0.8928	**0.9241**
Outdoor	WCR′7	0.9576	0.9503	0.8932	0.9312	0.949	0.9347	0.8928	0.9241
Outdoor	Distance	0.8482	0.7883	0.8047	0.8253	0.8253	0.7873	0.7646	0.7893
Outdoor	Google	0.6159	0.5152	0.4728	0.5376	0.5392	0.4540	0.4305	0.4777
Outdoor	Here	0.5555	0.4345	0.4794	0.4983	0.5040	0.4387	0.4704	0.4760

to WCR′4 could indicate the changes of the first three candidate rankings and the comparison of nDCG@5 under WCR′2 to WCR′7 could reflect the changes of first five candidate rankings. As can be observed, the score of nDCG@3 increased as *i* increased from 2 to 4, and the score of nDCG@3 remained the same after WCR′4. This is because when *i* equals 4, the nearest opposite-side candidate appears at 4, resulting in the first three candidates being unchanged. For the same reason, the score of nDCG@5 increased as *i* increased from 2 to 6 but remained the same from 6 to 7. Although the differences in nDCG@K for each WCR′i were limited by the existence of opposite-side candidates and the nearby street, these observed changes still reflected the validity of the proposed algorithm. The overall nDCG@3 and nDCG@5 performed the best when *i* reached at 6. Namely, the nearest opposite-side candidate was ranked in the sixth position. We chose $i = 6$ for the remaining work of this study.

The nDCG@K of Google and Here were systematically low due to their unclear ranking mechanism. Duplicate candidates generated by different interpolation methods and upper geographic level candidates (i.e., street range or city) were found in their output lists. Compared to the Distance method, WCR systematically improved nDCG@K scores. For the indoor inputs, WCR′6 improved by 3.70% and 4.78% for nDCG@3 and nDCG@5, respectively. For the outdoor inputs, WCR′6 improved by 12.83% and 17.08% for nDCG@3 and nDCG@5, respectively. For an outdoor scenario in a LDR land-use type, as shown in Figure 4.9 (a), WCR′6 ranked *208* at the sixth position, because it was located on the opposite side of the street, whereas the Distance method put *208* and *206* at the second and fourth position incorrectly. Figure 4.9 (b) depicts given an indoor input in a COM land-use type, WCR′6 delivered candidates ranked as: *1605, 1601, 1625, 514*, which are identical to the human-judged ranking. Our approach produced this ranking because *1605* has a higher weight than *1601*, as it was located inside the area of *1601*, whereas, *1625* and *514* were outside this area, receiving lower weights than *1601*. However, the Distance method gave 1625 and 514 the second and third positions based on their distances to the input. These differences illustrate the efficiency of our ranking approach and indicates the distance ranking approach leads to false-positive results more easily in an outdoor scenario. This finding helps to explain the better



(a)



(b)

Figure 4.9: Candidate ranking in indoor (a) and outdoor (b) scenarios

improvement for nDCG@K in the outdoor scenario than the indoor scenario. In terms of land-use types, our approach had a better performance for LDR and HDR types over COM type, resulting in the highest scores for LDR types. This can be explained by the fact that residential houses are distributed more regularly on the two sides of the streets; candidate selection can be efficiently constrained by the proposed spatial validations. In contrast, commercial buildings are typically clustered inside certain areas. Thus fewer candidates are constrained by *Street Side Validation*.

4.6.2.3 Impact of GPS Uncertainty

The third question concerns the efficiency of the proposed algorithm, which attempts to incorporate the uncertainty of input GPS into reverse geocoding output. It is worth mentioning that this algorithm aims to change the order of candidates to reflect GPS uncertainty instead of generating the most probable candidate by correcting GPS errors. Since the reverse geocoding workflow seldom considers GPS uncertainty, the standard assessment for this algorithm is hard to find in the existing literature. To conduct the evaluation, we collected real GPS coordinates and their accuracy readings by fixing phone positions. Specifically, at each land-use type, we fixed the phone in one indoor location and one outdoor location and recorded the readings of 10 GPS coordinates along with their accuracy readings. In total, we had 60 coordinates and the corresponding accuracy readings. We first examined whether GPS uncertainties could impact candidate rankings by applying GPS accuracy readings of 3, 10, and 30 meters, which were roughly 10th, 50th, and 90th percentiles among the collected readings. Figure 4.10 and Table 4.4 depict the output candidate list, given input coordinates with different accuracy readings (i.e., CEP radius). As one can see, when the CEP was relatively small (i.e., 3 meters), candidate lists remained unchanged. When the CEP became larger, opposite-site candidates (i.e., 210 or 212) joined the top five list, and their ranking positions increased. Such changes were more significant in the outdoor scenario than in the indoor scenario, as input coordinates move closer to the street centerline. When the CEP was 30 meters, *210* and *212* were ranked in the third and fifth positions in the outdoor scenario, whereas *210* appeared at the fourth position in the indoor scenario. Therefore, large GPS uncertainties were observed to impact the final candidate lists. We then processed all 60 GPS records and evaluated

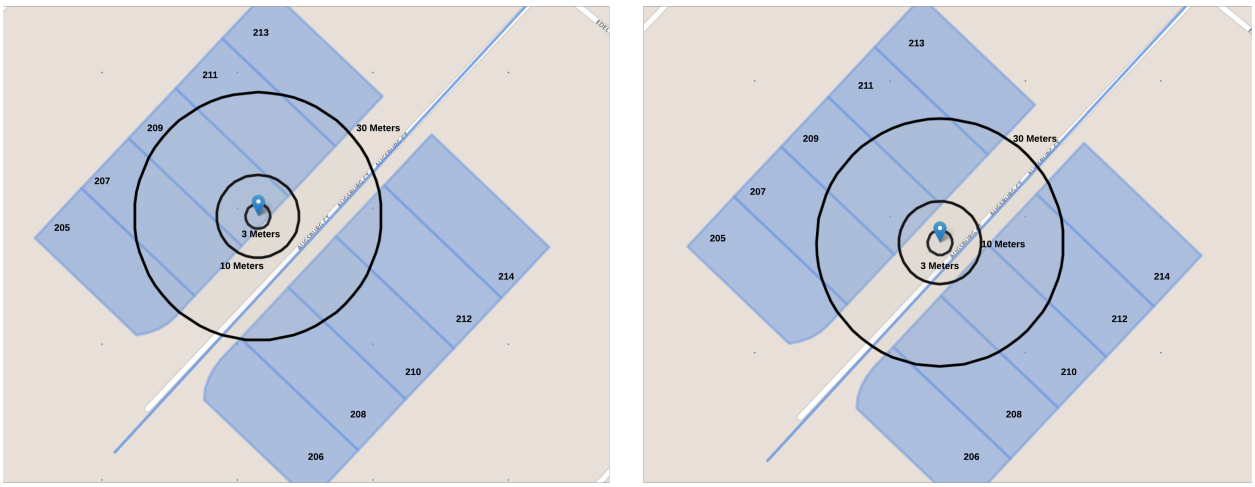


Figure 4.10: Input GPS coordinates with various accuracy in indoor (a) and outdoor (b) scenarios

outputs by humans. Table 4.5 summarizes the change rate of candidate ranking (i.e., new candidates join the top five list or the change in the order of the original top five candidates) with the consideration of GPS uncertainty to explore the impact from building distributions, input position (i.e., indoor or outdoor), and GPS accuracy (i.e., CEP). Overall, the ranking change was more significant for the outdoor coordinates, as the nearby opposite-side candidates were more likely to become top five candidates. Regarding different land-use types, LDR and HDR buildings were more likely to result in changes in candidate lists, whereas there were no such changes with respect to COM buildings, even though CEP was larger on the COM land-use type than others. This is because LDR and HDR buildings are more densely distributed than COM ones on the selected sites in this experiment. The minimal distance among the top five candidates was roughly 15 meters, 25 meters, and 150 meters on LDR, HDR, and COM land-use types, respectively. Finally, reviewers compared these two output candidate lists obtained by implementing and not implementing Algorithm 6 to judge the candidate ranking through figures that were similar to Figure 4.10. We quantified the agreement between the human-judged results and the system outputs by the percentage of agreement and the nDCG@5 score. In total, we found that the system output agreed with 93.3% of the votes of the human judges. Four out of the 60 sets of ranking lists were considered

Table 4.4: Candidate list based on various CEP radius under indoor and outdoor scenarios

Input Scenarios	CEP Radius			
	0 meter	3 meters	10 meters	30 meters
Indoor	209 Augsburg Ct.	209 Augsburg Ct.	209 Augsburg Ct.	209 Augsburg Ct.
	207 Augsburg Ct.	207 Augsburg Ct.	207 Augsburg Ct.	207 Augsburg Ct.
	211 Augsburg Ct.	211 Augsburg Ct.	211 Augsburg Ct.	211 Augsburg Ct.
	205 Augsburg Ct.	205 Augsburg Ct.	213 Augsburg Ct.	**210 Augsburg Ct.**
	213 Augsburg Ct.	213 Augsburg Ct.	**210 Augsburg Ct.**	213 Augsburg Ct.
Outdoor	209 Augsburg Ct.	209 Augsburg Ct.	209 Augsburg Ct.	209 Augsburg Ct.
	211 Augsburg Ct.	211 Augsburg Ct.	211 Augsburg Ct.	211 Augsburg Ct.
	207 Augsburg Ct.	207 Augsburg Ct.	207 Augsburg Ct.	**210 Augsburg Ct**.
	213 Augsburg Ct.	213 Augsburg Ct.	**210 Augsburg Ct.**	207 Augsburg Ct.
	205 Augsburg Ct.	205 Augsburg Ct.	**212 Augsburg Ct.**	**212 Augsburg Ct.**

discrepancies between the human and the algorithm. All four cases where the input coordinates were very close to the boundary of parcels on the LDR land-use type. Reviewers often determined that the last few candidates (i.e., the 4th or 5th candidates) should be the candidates that belong to other street blocks. We believe that returning such candidates may lead to practical problems, such as a driver needing to detour to incorrect street blocks. Such settings within our algorithm resulted in the average nDCG@5 of 60 inputs being 0.9824. Although there is no baseline for comparison, the experimental results presented above provide evidence that the proposed algorithm can successfully reflect the input GPS uncertainties.

4.7 Conclusions

In this chapter, we proposed a probabilistic framework for improving reverse geocoding output by delivering spatially close, topologically correct, and uncertainty-quantified candidates as well as incorporating input GPS uncertainties. First, we presented a reverse geocoding workflow that can adapt all existing address models and leverage spatial topology relationships. By comparing

Table 4.5: Candidate list change rates by considering GPS uncertainty under different scenarios

Input Scenario	Measurement	Land-use Type		
		LDR	HDR	COM
Indoor	CEP Range (meters)	2.31-12.80	2.36-11.79	2.23-15.00
Indoor	Change Rate (%)	20	0	0
Outdoor	CEP Range (meters)	6.43-19.30	3.13-35.38	16.08-49.31
Outdoor	Change Rate (%)	100	90	0

this approach with state-of-the-art commercial reverse geocoding systems, the proposed workflow was shown to be capable of producing a comparable amount of correct first candidates. Next, we introduced a scoring method to fully quantify each candidate in explicitly spatial terms (i.e., distance and topology) for each step in the proposed workflow to rank multiple candidates. The measured agreement between system output ranking and human-ordered ranking demonstrated the efficiency of the proposed ranking mechanism. Finally, we proposed an algorithm to propagate GPS uncertainty to final reverse geocoding output. The experimental results indicate that our algorithm can successfully propagate GPS uncertainty to its outputs. This work has attempted to deliver better reverse geocoding output by leveraging the pure spatial and geometric attributes of the reference data. We envision that more advanced candidate suggestion algorithms specific to different application scenarios can be built upon the quantification produced in this work.

5. SUMMARY AND FUTURE WORK

5.1 Summary

The dissertation is motivated by the wide adoption of geocoding and reverse geocoding systems as a data-processing procedure in various spatial analysis and application scenarios, which demands high output data quality in terms of match rate, spatial accuracy, and metadata descriptions when facing (low-quality) input. Research presented in this dissertation has systematically investigated various approaches to improve the output quality of geocoding and reverse geocoding systems.

Specifically, both geocoding and reverse geocoding workflow have been studied and decomposed into small sub-tasks. A geocoding workflow is split into two parts: text retrieval, which is responsible for retrieving address reference records from reference datasets based on input queries, and geocoding interpolation, which derives the final output coordinates on these retrieved address records. A reverse geocoding workflow is considered as retrieving nearest candidates to input coordinates and re-ranking these candidates by certain criteria. Chapter 2 and Chapter 3 enhance these two aforementioned parts of a geocoding workflow, respectively. Chapter 4 focuses on improving the output quality of reverse geocoding.

Chapter 2 benchmarks three sub-tasks inside the text retrieval process of a geocoding workflow: parsing, matching, and ranking. In this benchmark, (1) an automatic approach that analyzes human input patters from geocoding historic log data and utilizes such patterns to synthesize low-quality geocoding input. Using the synthesized address descriptions as testing input can reflect the performance of a geocoding systems that are closed to how it perform when facing user inputs in real scenarios; (2) an unified evaluation protocol including testing datasets, evaluation procedures and metrics is defined for these three sub-tasks; and (3) various parsing, matching, ranking techniques are assessed for low-quality geocoding input data, and experimental results suggest a set of methods that lead to the best performance for each geocoding text retrieval sub-task. Evaluation

results are expected to be reproducible to serve as solid baselines for future development.

In Chapter 3, a new interpolation method, which combines deep learning object detection technique with a typical geocoding workflow, is proposed to overcome drawbacks of typical geocoding interpolation methods and limitations of reference datasets. With the integration of the object detection technique into geocoding workflow, building rooftop centroid location becomes visible during the geocoding process, which helps to largely increase the output coordinate accuracy while maintaining comparable match rates compared to the original workflow. When original address reference data can pinpoint to correct parcels, the proposed approach outperformed a commercial geocoding solution in spatial accuracy. The proposed approach's performance is sensitive to building distributions on different land-use types (i.e., low-density residential, high-density residential, and commercial land-use types), suggesting such an approach should be utilized based on land-use types.

In Chapter 4, a probabilistic framework for improving reverse geocoding output is developed. Specifically, the proposed reverse geocoding work takes both distance and topological relationships into account to retrieve nearby address descriptions and ranks candidates by quantifying distance and topology a candidate has in relation to the input in a probabilistic manner. Experimental results show that the first candidate's correctness is improved compared to other commercial reverse geocoding solutions, and the method using distance as the sole metric to rank candidates. Meanwhile, the overall ranking quality is found to be closed to human spatial recognition via a human ranking evaluation. To accommodate the inevitable input GPS errors, the proposed approach can propagate such GPS errors through the reverse geocoding workflow and are reflected on the output candidate ranking. The efficiency of GPS error propagation is proven by comparing changes of output candidate ranking under different combinations of GPS error readings and input locations.

5.2 Future Work

The work presented in this dissertation demonstrates how we improve the output quality of geocoding and reverse geocoding systems when facing low-quality input. There is still plenty of additional research opportunities that can be done from our work.

5.2.1 Geocoding

For the geocoding text retrieval process, future work avenues can be (1) building a universal address parser. Recent advance in word embedding offers the opportunity to convert word in different languages into dense vector representations. [126] can be seen as the first step towards this direction. Moving forward, different neural network architectures and different ways to utilize pre-trained word embedding models can be evaluated. (2) moving to scenario-based geocoding text retrieval methods. In Chapter 2, geocoding transactions that are used to mining human input patterns are relatively limited in terms of the number of transactions and time duration that transactions happen. Meanwhile, we have not considered probabilities of errors that occur when injecting errors into address descriptions. In the future, long-term geocoding transactions can be used to better capture how likely an error occurs under different usage scenarios. Under this direction, the selected text retrieval methods (i.e., parsing, matching, and ranking) can be suitable to particular input patterns and, eventually, make geocoding systems adaptive to different application scenarios.

For the geocoding interpolation methods, since the address point-based interpolation method yields the highest spatial accuracy output, one future direction can be using the presented workflow proposed in Chapter 3 to update the address reference dataset by geocoding every address description in reference datasets. Meanwhile, since the output geocodes are generated from the presented workflow are observed to varies on building distributions, and land-use types, the method to generate rooftop centroid should be adaptive to these variants on buildings and land usages such as single building and multi-building complex.

Additionally, one promising perspective of future geocoding system enhancement is to create continuously self-improving geocoding systems. Such a concept was first proposed by [59], and the self-improving capability was gained by keeping indexing address descriptions with errors. In Chapter 2, we show that various approaches can be utilized to improve the quality of geocoding output when facing low-quality input. Namely, synthesized low-quality input can be used to train a neural network-based address parser and to distill suitable fuzzy matching and candidate ranking approaches. In Chapter 3, we demonstrate one way to update coordinates of address point reference

data, improving the spatial accuracy of reference datasets. By integrating these two features into a typical geocoding workflow, geocoding systems are expected to have the capability to learn from historic geocoding data. Specifically, geocoding input and the corresponded output data will be analyzed to identify reasons that lead to low-quality output and trigger different approaches to enhance the system itself. For instance, updating reference data coordinates can be activated (1) if retrieved candidates have relatively high textual similarity but have low spatial accuracy or (2) to accommodate the reference map coordinates changes resulted from the plate tectonic motion. Meanwhile, input address descriptions can be used as a training sample to learn how does such input errors and variants occur and what is the robust text retrieval strategy to handle such errors, once geocoding input cannot be resolved as a matching status or the first few candidates have relatively low textual similarity with respect to the input. Given the uncertainty of user input, the feasibility of such a self-improving geocoding system concept is worthy of further investigation.

5.2.2 Reverse geocoding

Reverse geocoding systems can be improved in various ways. In terms of output candidate ranking by reverse geocoding workflow, we assumed that the reference datasets used did not introduce errors or uncertainties in the reverse geocoding workflow, which is not likely true in reality [11]. The spatial accuracy and coverage of reference datasets were observed to impact the selection of the first candidate and the candidate rankings. Therefore, further quantification of reference data uncertainties could help to formalize error prediction models for reverse geocoding output, similar to the geocoding process [26]. In terms of output candidate ranking using auxiliary information, more factors can be incorporated and quantified. For example, time [106] has been used to differentiate the ranking of POI categories. As for low-quality coordinate input, we only considered that input obtained through Android systems. However, uncertainty coming from other signal sources such as GNSS, WiFi, or cell phone signal could be considered and evaluated separately. Another direction could investigate system performance (i.e., system throughput) enhancement by utilizing novel spatial indexing and data storage techniques or more efficient calculation approaches.

REFERENCES

[1] Z. Yin, C. Zhang, D. W. Goldberg, and S. Prasad, "An nlp-based question answering framework for spatio-temporal analysis and visualization," in *Proceedings of the 2019 2nd International Conference on Geoinformatics and Data Analysis*, pp. 61–65, ACM, 2019.

[2] A. T. Murray, T. H. Grubesic, R. Wei, and E. A. Mack, "A hybrid geocoding methodology for spatio-temporal data," *Transactions in GIS*, vol. 15, no. 6, pp. 795–809, 2011.

[3] K. M. Rose, J. L. Wood, S. Knowles, R. A. Pollitt, E. A. Whitsel, A. V. D. Roux, D. Yoon, and G. Heiss, "Historical measures of social context in life course studies: retrospective linkage of addresses to decennial censuses," *International journal of health geographics*, vol. 3, no. 1, p. 27, 2004.

[4] G. Rushton, M. P. Armstrong, J. Gittler, B. R. Greene, C. E. Pavlik, M. M. West, and D. L. Zimmerman, "Geocoding in cancer research: a review," *American journal of preventive medicine*, vol. 30, no. 2, pp. S16–S24, 2006.

[5] K. S. McLeod, "Our sense of snow: the myth of john snow in medical geography," *Social science & medicine*, vol. 50, no. 7-8, pp. 923–935, 2000.

[6] K. Zinszer, C. Jauvin, A. Verma, L. Bedard, R. Allard, K. Schwartzman, L. de Montigny, K. Charland, and D. L. Buckeridge, "Residential address errors in public health surveillance data: A description and analysis of the impact on geocoding," *Spatial and Spatio-temporal Epidemiology*, vol. 1, no. 2-3, pp. 163–168, 2010.

[7] Y. J. McDonald, D. W. Goldberg, I. C. Scarinci, P. E. Castle, J. Cuzick, M. Robertson, and C. M. Wheeler, "Health service accessibility and risk in cervical cancer prevention: Comparing rural versus nonrural residence in new mexico," *The Journal of Rural Health*, no. 4, pp. 382–392, 2017.

[8] D. Li, "Geocoding and reverse geocoding," in *Comprehensive Geographic Information Systems* (B. Huang, ed.), pp. 95–109, Oxford: Elsevier, 2018.

[9] C. D. Smith and J. Mennis, "Incorporating geographic information science and technology in response to the covid-19 pandemic," *Preventing Chronic Disease*, vol. 17, 2020.

[10] U. Qazi, M. Imran, and F. Ofli, "Geocov19: a dataset of hundreds of millions of multilingual covid-19 tweets with location information," *SIGSPATIAL Special*, vol. 12, no. 1, pp. 6–15, 2020.

[11] P. A. Zandbergen, "Geocoding quality and implications for spatial analysis," *Geography Compass*, vol. 3, no. 2, pp. 647–680, 2009.

[12] D. W. Goldberg, G. M. Jacquez, and N. Mullan, "Geocoding and health," in *Geographic Health Data: Fundamental Techniques for Analysis*, pp. 51–71, CABI, 2013.

[13] G. M. Jacquez, "A research agenda: does geocoding positional error matter in health gis studies?," *Spatial and spatio-temporal epidemiology*, vol. 3, no. 1, pp. 7–16, 2012.

[14] D. Nuvolone, R. della Maggiore, S. Maio, R. Fresco, S. Baldacci, L. Carrozzi, F. Pistelli, and G. Viegi, "Geographical information system and environmental epidemiology: a cross-sectional spatial analysis of the effects of traffic-related air pollution on population respiratory health," *Environmental Health*, vol. 10, no. 1, p. 12, 2011.

[15] P. A. Zandbergen and J. W. Green, "Error and bias in determining exposure potential of children at school locations using proximity-based gis techniques," *Environmental Health Perspectives*, vol. 115, no. 9, p. 1363, 2007.

[16] E. P. Washburn, M. J. Orza, J. A. Berlin, W. J. Nicholson, A. C. Todd, H. Frumkin, and T. C. Chalmers, "Residential proximity to electricity transmission and distribution equipment and risk of childhood leukemia, childhood lymphoma, and childhood nervous system tumors: systematic review, evaluation, and meta-analysis," *Cancer Causes & Control*, vol. 5, no. 4, pp. 299–309, 1994.

[17] Á. Briz-Redón, F. Martinez-Ruiz, and F. Montes, "Reestimating a minimum acceptable geocoding hit rate for conducting a spatial analysis," *International Journal of Geographical Information Science*, vol. 34, no. 7, pp. 1283–1305, 2020.

[18] D. W. Goldberg and M. G. Cockburn, "Improving geocode accuracy with candidate selection criteria," *Transactions in GIS*, vol. 14, no. s1, pp. 149–176, 2010.

[19] D. W. Goldberg, M. Ballard, J. H. Boyd, N. Mullan, C. Garfield, D. Rosman, A. M. Ferrante, and J. B. Semmens, "An evaluation framework for comparing geocoding systems," *International journal of health geographics*, vol. 12, no. 1, p. 50, 2013.

[20] D. Goldberg, "Geocoding techniques and technologies for location-based services," in *Advanced Location-Based Technologies and Services*, pp. 75–106, CRC Press: Boca Raton, FL, 2013.

[21] P. Christen, D. Belacic, *et al.*, "Automated probabilistic address standardisation and verification," in *Australasian Data Mining Conference*, Citeseer, 2005.

[22] S. Mokhtari, A. Mahmoody, D. Yankov, and N. Xie, "Tagging address queries in maps search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 9547–9551, 2019.

[23] L. Li, W. Wang, B. He, and Y. Zhang, "A hybrid method for chinese address segmentation," *International Journal of Geographical Information Science*, vol. 32, no. 1, pp. 30–48, 2018.

[24] B. Ranzijn, "A geocoding algorithm based on a comparative study of address matching techniques," 2013.

[25] Y. Lin, M. Kang, Y. Wu, Q. Du, and T. Liu, "A deep learning architecture for semantic address matching," *International Journal of Geographical Information Science*, vol. 34, no. 3, pp. 559–576, 2020.

[26] D. W. Goldberg and M. G. Cockburn, "Improving geocode accuracy with candidate selection criteria," *Transactions in GIS*, vol. 14, no. s1, pp. 149–176, 2010.

[27] A. T. Murray, T. H. Grubesic, R. Wei, and E. A. Mack, "A hybrid geocoding methodology for spatio-temporal data," *Transactions in GIS*, vol. 15, no. 6, pp. 795–809, 2011.

[28] D. W. Goldberg, "A geocoding best practices guide," 2008.

[29] M. R. Cayo and T. O. Talbot, "Positional error in automated geocoding of residential addresses," *International Journal of Health Geographics*, vol. 2, p. 10, Dec 2003.

[30] R. Bakshi, C. A. Knoblock, and S. Thakkar, "Exploiting online sources to accurately geocode addresses," in *Proceedings of the 12th Annual ACM International Workshop on Geographic Information Systems*, GIS '04, (New York, NY, USA), pp. 194–203, ACM, 2004.

[31] C. A. Knoblock, A. R. Joshi, A. Megotia, M. Pham, and C. Ursaner, "Automatic spatio-temporal indexing to integrate and analyze the data of an organization," in *Proceedings of the 3rd ACM SIGSPATIAL Workshop on Smart Cities and Urban Analytics*, p. 7, ACM, 2017.

[32] P. Christen, T. Churches, *et al.*, "Febrl-freely extensible biomedical record linkage," 2002.

[33] P. A. Zandbergen, T. C. Hart, K. E. Lenzer, and M. E. Camponovo, "Error propagation models to examine the effects of geocoding quality on spatial analysis of individual-level datasets.," *Spatial and spatio-temporal epidemiology*, vol. 3 1, pp. 69–82, 2012.

[34] N. B. Ngo and R. N. Owen, "Expediting reverse geocoding with a bounding region," Feb. 25 2014. US Patent 8,660,793.

[35] M. Ma, Z. Zhong, N. Guo, N. Jing, and W. Xiong, "An efficient reverse geocoding method based on global subdivision model," in *Geoinformatics, 2016 24th International Conference on*, pp. 1–9, IEEE, 2016.

[36] K. R. Searight, D. J. Logan, J. B. I. Freddie, C. J. Loher, and B. R. Charlton, "Reverse geocoding system using combined street segment and point datasets," Feb. 23 2010. US Patent 7,668,651.

[37] C. Hauff, "A study on the accuracy of flickr's geotag data," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pp. 1037–1040, ACM, 2013.

[38] M. Ye, K. Janowicz, C. Mülligann, and W.-C. Lee, "What you are is when you are: the temporal dimension of feature types in location-based social networks," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 102–111, ACM, 2011.

[39] S. Yu, S. Spaccapietra, N. Cullot, and M.-A. Aufaure, "User profiles in location-based services: Make humans more nomadic and personalized," in *Proc. of the International Workshop on Next Generation Geospatial Information*, no. LBD-CONF-2003-009 in NG2I 2003, pp. 1–6, 2003.

[40] H. Chen, M. S. Arefin, Z. Chen, and Y. Morimoto, "Place recommendation based on users check-in history for location-based services," *International Journal of Networking and Computing*, vol. 3, no. 2, pp. 228–243, 2013.

[41] V. Vieira, A. Fraser, T. Webster, G. Howard, and S. Bartell, "Accuracy of automated and e911 geocoding methods for rural addresses," *Epidemiology*, vol. 19, no. 6, p. S352, 2008.

[42] D. W. Goldberg and M. G. Cockburn, "The effect of administrative boundaries and geocoding error on cancer rates in california," *Spatial and Spatio-temporal Epidemiology*, vol. 3, no. 1, pp. 39 – 54, 2012. Special Issue on Geocoding in the Health Sciences.

[43] D. W. Goldberg, J. P. Wilson, and C. A. Knoblock, "From text to geographic coordinates: the current state of geocoding," *URISA-WASHINGTON DC-*, vol. 19, no. 1, p. 33, 2007.

[44] T. H. Grubesic and A. T. Murray, "Assessing positional uncertainty in geocoded data," in *Proceedings of the 24th urban data management symposium*, 2004.

[45] S. E. Hurley, T. M. Saunders, R. Nivas, A. Hertz, and P. Reynolds, "Post office box addresses: a challenge for geographic information system-based studies," *Epidemiology*, vol. 14, no. 4, pp. 386–391, 2003.

[46] Z. Yin, A. Ma, and D. W. Goldberg, "A deep learning approach for rooftop geocoding," *Transactions in GIS*, vol. 23, no. 3, pp. 495–514, 2019.

[47] N. Kirielle, P. Christen, and T. Ranbaduge, "Outlier detection based accurate geocoding of historical addresses," in *Australasian Conference on Data Mining*, pp. 41–53, Springer, 2019.

[48] P. Christen, *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer Science & Business Media, 2012.

[49] P. A. Zandbergen, "A comparison of address point, parcel and street geocoding techniques," *Computers, Environment and Urban Systems*, vol. 32, no. 3, pp. 214–232, 2008.

[50] V. Sengar, T. Joshi, J. Joy, S. Prakash, and K. Toyama, "Robust location search from text queries," in *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, p. 24, ACM, 2007.

[51] M. J. Hutchinson, *Developing an agent-based framework for intelligent geocoding*. PhD thesis, Curtin University, 2010.

[52] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147, 2003.

[53] M. Van Erp, P. Mendes, H. Paulheim, F. Ilievski, J. Plu, G. Rizzo, and J. Waitelonis, "Evaluating entity linking: An analysis of current benchmark datasets and a roadmap for doing a better job," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pp. 4373–4379, 2016.

[54] A. Jimeno, E. Jimenez-Ruiz, V. Lee, S. Gaudan, R. Berlanga, and D. Rebholz-Schuhmann, "Assessment of disease named entity recognition on a corpus of annotated sentences," in *BMC bioinformatics*, vol. 9, p. S3, BioMed Central, 2008.

[55] J. Wang and Y. Hu, "Enhancing spatial and textual analysis with eupeg: an extensible and unified platform for evaluating geoparsers," *Transactions in GIS*, vol. 23, no. 6, pp. 1393–1419, 2019.

[56] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[57] R. Nogueira, W. Yang, K. Cho, and J. Lin, "Multi-stage document ranking with bert," *arXiv preprint arXiv:1910.14424*, 2019.

[58] W. W. Cohen, P. Ravikumar, S. E. Fienberg, *et al.*, "A comparison of string distance metrics for name-matching tasks.," in *IIWeb*, vol. 2003, pp. 73–78, 2003.

[59] K. Clemens, "Geocoding user queries," 2020.

[60] K. Clemens, "Geocoding with openstreetmap data," *GEOProcessing 2015*, p. 10, 2015.

[61] L. Gu, R. Baxter, D. Vickers, and C. Rainsford, "Record linkage: Current practice and future directions," *CSIRO Mathematical and Information Sciences Technical Report*, vol. 3, p. 83, 2003.

[62] P. Christen, "A survey of indexing techniques for scalable record linkage and deduplication," *IEEE transactions on knowledge and data engineering*, vol. 24, no. 9, pp. 1537–1555, 2011.

[63] D. W. Goldberg, "Improving geocoding match rates with spatially-varying block metrics," *Trans. GIS*, vol. 15, pp. 829–850, 2011.

[64] G. Dias and T. Honkela, "Term weighting in short documents for document categorization , keyword extraction and query expansion," 2012.

[65] P. Christen, "A comparison of personal name matching: Techniques and practical issues," in *Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06)*, pp. 290–294, IEEE, 2006.

[66] L. Philips, "The double metaphone search algorithm," *C/C++ users journal*, vol. 18, no. 6, pp. 38–43, 2000.

[67] H. Keskustalo, A. Pirkola, K. Visala, E. Leppänen, and K. Järvelin, "Non-adjacent digrams improve matching of cross-lingual spelling variants," in *International symposium on string processing and information retrieval*, pp. 252–265, Springer, 2003.

[68] C. Li, J. Lu, and Y. Lu, "Efficient merging and filtering algorithms for approximate string searches," in *2008 IEEE 24th International Conference on Data Engineering*, pp. 257–266, IEEE, 2008.

[69] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, *et al.*, "Okapi at trec-3," *Nist Special Publication Sp*, vol. 109, p. 109, 1995.

[70] D. W. Goldberg, "Improving geocoding match rates with spatially-varying block metrics," *Transactions in GIS*, vol. 15, no. 6, pp. 829–850, 2011.

[71] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[72] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242, pp. 29–48, Citeseer, 2003.

[73] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," *arXiv preprint arXiv:1603.01360*, 2016.

[74] J. Zhu, Y. Xia, L. Wu, D. He, T. Qin, W. Zhou, H. Li, and T.-Y. Liu, "Incorporating bert into neural machine translation," *arXiv preprint arXiv:2002.06823*, 2020.

[75] L. Ge and T.-S. Moh, "Improving text classification with word embedding," in *2017 IEEE International Conference on Big Data (Big Data)*, pp. 1796–1805, IEEE, 2017.

[76] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[77] Z. A. Yilmaz, S. Wang, W. Yang, H. Zhang, and J. Lin, "Applying bert to document retrieval with birch," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pp. 19–24, 2019.

[78] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[79] A. Patel, A. Sands, C. Callison-Burch, and M. Apidianaki, "Magnitude: A fast, efficient universal vector embedding utility package," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, (Brussels, Belgium), pp. 120–126, Association for Computational Linguistics, Nov. 2018.

[80] X. Yang, D. Lo, X. Xia, L. Bao, and J. Sun, "Combining word embedding with information retrieval to recommend similar bug reports," in *2016 IEEE 27Th international symposium on software reliability engineering (ISSRE)*, pp. 127–137, IEEE, 2016.

[81] M. Wang, V. Haberland, A. Yeo, A. Martin, J. Howroyd, and J. M. Bishop, "A probabilistic address parser using conditional random fields and stochastic regular grammar," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pp. 225–232, IEEE, 2016.

[82] H. Craig, D. Yankov, R. Wang, P. Berkhin, and W. Wu, "Scaling address parsing sequence models through active learning," in *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 424–427, 2019.

[83] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition, corr abs/1812.09449 (2018)," *arXiv preprint arXiv:1812.09449*, 2018.

[84] N. Reimers and I. Gurevych, "Optimal hyperparameters for deep lstm-networks for sequence labeling tasks," *arXiv preprint arXiv:1707.06799*, 2017.

[85] A. Aghaebrahimian and M. Cieliebak, "Hyperparameter tuning for deep learning in natural language processing," in *4th swiss text analytics conference (swisstext 2019), winterthur, june 18-19 2019*, Swisstext, 2019.

[86] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[87] N. Reimers and I. Gurevych, "Alternative weighting schemes for elmo embeddings," *arXiv preprint arXiv:1904.02954*, 2019.

[88] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," *arXiv preprint arXiv:1603.01354*, 2016.

[89] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," *arXiv preprint arXiv:1603.01360*, 2016.

[90] N. Reimers and I. Gurevych, "Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Copenhagen, Denmark), pp. 338–348, 09 2017.

[91] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," *arXiv preprint arXiv:1603.01360*, 2016.

[92] F. Boscoe, "The science and art of geocoding: tips for improving match rates and handling unmatched cases in analysis," in *Geocoding Health Data: The Use of Geographic Codes in Cancer Prevention and Control, Research and Practice*, pp. 95–110, CRC Press: Boca Raton, FL, 2008.

[93] G. Rushton, M. P. Armstrong, J. Gittler, B. R. Greene, C. E. Pavlik, M. M. West, and D. L. Zimmerman, "Geocoding in cancer research: a review," *American journal of preventive medicine*, vol. 30, no. 2, pp. S16–S24, 2006.

[94] X. Chen, S. Xiang, C.-L. Liu, and C.-H. Pan, "Vehicle detection in satellite images by hybrid deep convolutional neural networks," *IEEE Geoscience and remote sensing letters*, vol. 11, no. 10, pp. 1797–1801, 2014.

[95] T.-B. Xu, G.-L. Cheng, J. Yang, and C.-L. Liu, "Fast aircraft detection using end-to-end fully convolutional network," in *Digital Signal Processing (DSP), 2016 IEEE International Conference on*, pp. 139–143, IEEE, 2016.

[96] F. Chen, R. Ren, T. Van de Voorde, W. Xu, G. Zhou, and Y. Zhou, "Fast automatic airport detection in remote sensing images using convolutional neural networks," *Remote Sensing*, vol. 10, no. 3, p. 443, 2018.

[97] E. Delmelle, W. Tang, M. Zheng, Y. Lan, and C. Owusu, "Geocoding fundamentals and associated challenges," in *Geospatial Data Science Techniques and Applications*, pp. 57–78, CRC Press, 2017.

[98] A. Schultz, K. Beyer, and G. Rushton, "Using ZIP® codes as geocodes in cancer research," in *Geocoding Health Data*, pp. 37–67, CRC Press, nov 2007.

[99] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[100] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[101] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *European conference on computer vision*, pp. 346–361, Springer, 2014.

[102] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[103] D. L. Zimmerman and J. Li, "The effects of local street network characteristics on the positional accuracy of automated geocoding for geographic health studies," *International journal of health geographics*, vol. 9, no. 1, p. 10, 2010.

[104] D. L. Zimmerman, X. Fang, S. Mazumdar, and G. Rushton, "Modeling the probability distribution of positional errors incurred by residential address geocoding," *International Journal of Health Geographics*, vol. 6, no. 1, p. 1, 2007.

[105] D. Roongpiboonsopit and H. A. Karimi, "Quality assessment of online street and rooftop geocoding services," *Cartography and Geographic Information Science*, vol. 37, no. 4, pp. 301–318, 2010.

[106] G. McKenzie and K. Janowicz, "Where is also about time: A location-distortion model to improve reverse geocoding using behavior-driven temporal semantic signatures," *Computers, Environment and Urban Systems*, vol. 54, pp. 1–13, 2015.

[107] J. S. Brownstein, C. Cassa, I. S. Kohane, and K. D. Mandl, "Reverse geocoding: concerns about patient confidentiality in the display of geospatial health data," in *AMIA Annual Symposium Proceedings*, vol. 2005, p. 905, American Medical Informatics Association, 2005.

[108] A. Podor, "Usability study on different visualization methods of crime maps.," *International Journal of Geoinformatics*, vol. 11, no. 4, 2015.

[109] C. Zhang, Z. Yin, P. Gao, and S. Prasad, "A visual analytics approach to exploration of hotels in overlaid drive-time polygons of attractions," in *International Symposium on Web and Wireless Geographical Information Systems*, pp. 28–40, Springer, 2019.

[110] S. Tiwari, S. Kaushik, P. Jagwani, and S. Tiwari, "A survey on lbs: system architecture, trends and broad research areas," in *International Workshop on Databases in Networked Information Systems*, pp. 223–241, Springer, 2011.

[111] M. Lv, L. Chen, Z. Xu, Y. Li, and G. Chen, "The discovery of personally semantic places based on trajectory data mining," *Neurocomputing*, vol. 173, pp. 1142–1153, 2016.

[112] Y. Tawk, P. Tomé, C. Botteron, Y. Stebler, and P.-A. Farine, "Implementation and performance of a gps/ins tightly coupled assisted pll architecture using mems inertial sensors," *Sensors (Basel, Switzerland)*, vol. 14, no. 2, p. 3768—3796, 2014.

[113] P. A. Zandbergen, "Accuracy of iphone locations: A comparison of assisted gps, wifi and cellular positioning," *Transactions in GIS*, vol. 13, no. s1, pp. 5–25, 2009.

[114] R. K. Huang and P. Piemonte, "Context-based reverse geocoding," July 29 2014. US Patent 8,792,917.

[115] B. Yan, K. Janowicz, G. Mai, and S. Gao, "From itdl to place2vec–reasoning about place type similarity and relatedness by learning embeddings from augmented spatial contexts," *Proceedings of SIGSPATIAL*, vol. 17, pp. 7–10, 2017.

[116] O. Kounadi, T. J. Lampoltshammer, M. Leitner, and T. Heistracher, "Accuracy and privacy aspects in free online reverse geocoding services," *Cartography and Geographic Information Science*, vol. 40, no. 2, pp. 140–153, 2013.

[117] D. L. Zimmerman, G. Rushton, M. P. Armstrong, J. Gittler, B. R. Greene, C. E. Pavlik, and M. M. West, *Geocoding health data: the use of geographic codes in cancer prevention and control, research and practice*. CRC Press, 2007.

[118] M. J. Egenhofer and R. D. Franzosa, "Point-set topological spatial relations," *International Journal of Geographical Information System*, vol. 5, no. 2, pp. 161–174, 1991.

[119] D. W. Goldberg, J. P. Wilson, and M. G. Cockburn, "Toward quantitative geocode accuracy metrics," in *ninth international symposium on spatial accuracy assessment in natural resources and environmental sciences*, pp. 329–32, 2010.

[120] R. Rogers, J. Lombardo, Z. Mednieks, and B. Meike, *Android application development: Programming with the Google SDK*. O'Reilly Media, Inc., 2009.

[121] P. A. Zandbergen, "Positional accuracy of spatial data: Non-normal distributions and a critique of the national standard for spatial data accuracy," *Transactions in GIS*, vol. 12, no. 1, pp. 103–130, 2008.

[122] D. T. Bui, C. T. Tran, B. Pradhan, I. Revhaug, and R. Seidu, "igeotrans – a novel ios application for gps positioning in geosciences," *Geocarto International*, vol. 30, no. 2, pp. 202–217, 2015.

[123] H. Zade, M. Drouhard, B. Chinh, L. Gan, and C. Aragon, "Conceptualizing disagreement in qualitative coding," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pp. 1–11, 2018.

[124] C. Zhang, Y. Chen, J. Yang, and Z. Yin, "An association rule based approach to reducing visual clutter in parallel sets," *Visual Informatics*, vol. 3, no. 1, pp. 48–57, 2019.

[125] W. Niu, Z. Liu, and J. Caverlee, "On local expert discovery via geo-located crowds, queries, and candidates," *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, vol. 2, no. 4, p. 14, 2016.

[126] M. Yassine, D. Beauchemin, F. Laviolette, and L. Lamontagne, "Leveraging subword embeddings for multinational address parsing," in *2020 6th IEEE Congress on Information Science and Technology (CiSt)*, pp. 353–360, IEEE, 2021.